# DP 3


Good Evening

## Today's Agenda

* Fractional Knapsack
* 0/1 Knapsack
* Unbounded Knapsack

# Fractional KnapSack

Given N cakes with their happiness & weight. Find max total happiness that can be kept in a bag with capacity = W.

Note → Cakes can be divided into pastries

N = 5

cap = 40

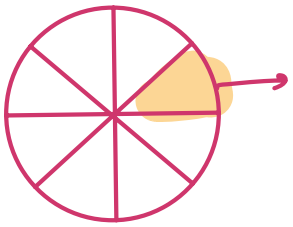
happiness [ ] = { 3 8 10 2 5 }

weight [ ] = { 10 4 20 8 15 }

Ideal → Pick the cake with max happiness

$\Sigma h = 10 + 8 + 5 + 0.3 \Rightarrow 23.3$

$\Sigma W = 20 + 4 + 15 + 1$



10 kg → 3 happinness

1 kg → $\frac{3}{10}$ unit = 0.3

N = 5

Cap = 40

h[] = { 4 8 10 2 5 }

wt[] = { 4 4 20 8 16 }

$\Sigma h = 10 + 8 + 5 \Rightarrow \underline{23}$

$\Sigma wt = 20 + 4 + 16$

* Greedy parameter → pick max happiness is not working

* Idea 2 → Pick the ele with maximum $\frac{\text{happiness}}{\text{weight}}$

h[] = { 4 8 10 2 5 }

wt[] = { 4 4 20 8 16 }

hp/wt = 1 2 0.5 0.25 0.3125

4 pastries

4 pastries

We need to pick 40 pastries

$\Sigma h = 8 + 4 + 10 + 3.75$ - 25.75

$\Sigma wt = 4 + 4 + 20 + 12$ - 40 kg

* Code

01. Sort the cakes (in descending) on the basis of $\frac{\text{happiness}}{\text{weight}}$ ratio

02.
```
for (i=0; i<n; i++){
    if (wt[i] ≤ capacity)
        ans += h[i];
        capacity -= wt[i];
    else{
        ans += ( h[i]/wt[i] * capacity )
        break;
    }
}
return ans;
```

TC : O(N log n)

SC : O(n)

# 0/1 KnapSack

Given N items each with a weight & value, find max value which can be obtained by picking items such that total weight of all items $\leq K$

Note 1 :- Every item can be picked at max 1 time

Note 2 :- We can not take a part of item

Eg:-

N = 4 items  K = 50

| Items: | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| Value[]: | 100 | 60 | 120 | 150 |
| Weight[]: | 20 | 10 | 30 | 40 |
| V/W = | 5 | 6 | 4 | 3.75 |

Greedy → Pick item with max V/W ratio ✗

$\Sigma V = 60 + 100 = 160$

$\Sigma wt = 10 + 20$

Greedy - Pick item with max value

$\Sigma V = 150 + 60 = 210$

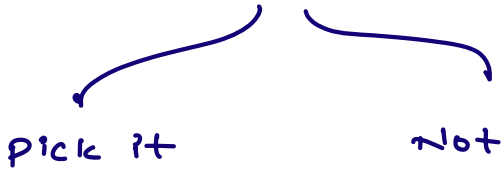
$\Sigma wt = 40 + 10 = 50$

* Correct ans $\quad \Sigma v = 100 + 120 = 220$

$\Sigma wt = 20 + 30 = 50$

Idea 2 → Generate all combinations.

Every ele has two choices



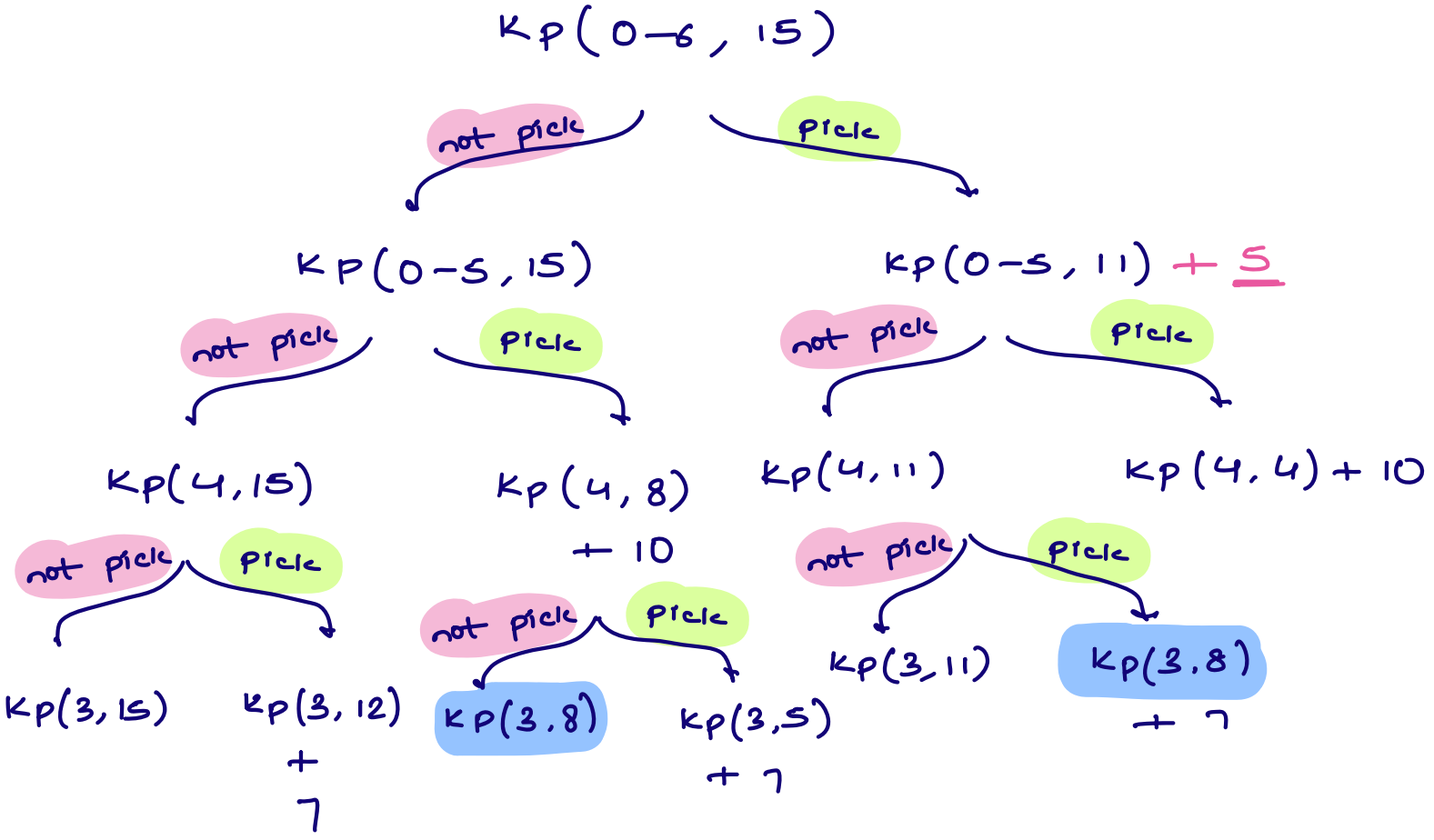

& return the ans which is providing the max value.

Cap = 15

N = 7

wt[] = { 4 1 5 4 3 7 4 }
0 1 2 3 4 5 6

val[] = { 3 2 8 3 7 10 5 }




# Code →

n-1 K

```
int knapSack ( int i, int j , int [] wt, int [] val )
{
  if ( i<0 || j≤0 ) return -∞;
  int notpick = knapsack ( i-1, j , wt, val );
  int pick = 0
  if ( j ≥ wt[i] ){
    int pick = knapsack ( i-1, j - wt[i] , w, val ) + val[i]
  }
  return max ( not pick, pick )
}
```

Top down

```
int [][] dp = new int[n][k+1];
```

```
int knapsack ( i , j , wt , val )
{
    if (i<0 || j≤0) return 0;

    if ( dp[i][j] != -1) return dp[i][j];

    int val = knapsack ( i-1, j , wt, val)

    if ( j ≥ wt)
    {
        val = max( val, knapsack( i-1, j-wt[i] , wt, val) +val[i])
    }

    return dp[i][j] = val;
}
```

```
int [] wt = {3  6  5  2  4}        cap=7
            0  1  2  3  4
int [] val = {12  20  15  6  10}
```

```
int dp[n+1] [k+1]
```

## * Bottom Up Approach

| val | wt |
|---|---|
| 12 | 3 |
| 20 | 6 |
| 15 | 5 |
| 6 | 2 |
| 10 | 4 |

| ele ↓ / cap → | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 12 | 12 | 12 | 12 | 12 |
| 2 | 0 | 0 | 0 | 12 | 12 | 12 | 20 | 20 |
| 3 | 0 | 0 | 0 | 12 | 12 | 15 | 20 | 20 |
| 4 | 0 | 0 | 6 | 12 | 12 | 18 | 20 | 21 |
| 5 | 0 | 0 | 6 | 12 | 12 | 18 | 20 | 22 |

not pick 4th ele → we have to ask our first 3 ele to go & fill the bag of cap = 3 & generate max value

⇒ 12

pick 4th ele = 6 + ram bag cap = 1

ask 3 ele to go & fill the bag of cap=1 & gen max value

dp[i][j] = max value that we can generate using i elements & j as bag capacity

$$dp[i][j] = \overset{\text{Max}}{\Big(} dp[i-1][j],\ val[i-1] + dp[i-1][j-wt[i-1]] \Big)$$

$dp[i-1][j]$ ↑ not pick

$val[i-1] + dp[i-1][j-wt[i-1]]$ underbrace: if $(j - wt[i-1] \geq 0)$

TC : O(n * k)

SC : O(n * k)

```
* for (i=0; i<n; i++)
  |  for (j=0; j<m; j++){
  |  |  if (i==0 || j==0) dp[i][j]=0
  |  |  else{
  |  |  |  rej = dp[i-1][j]
  |  |  |  sel = 0
  |  |  |  if ( j ≥ wt[i-1])
  |  |  |  |  select = val[i-1] + dp[i-1][j- wt[i-1]];
  |  |  |  3
  |  |  |  dp[i][j]= max( rej, select);
  |  |  3
  |  2
  3
  return dp[N][K];
```

# Unbounded Knapsack

Given N items each with a weight & value, find max value which can be obtained by picking items such that total weight of all items $\leq K$

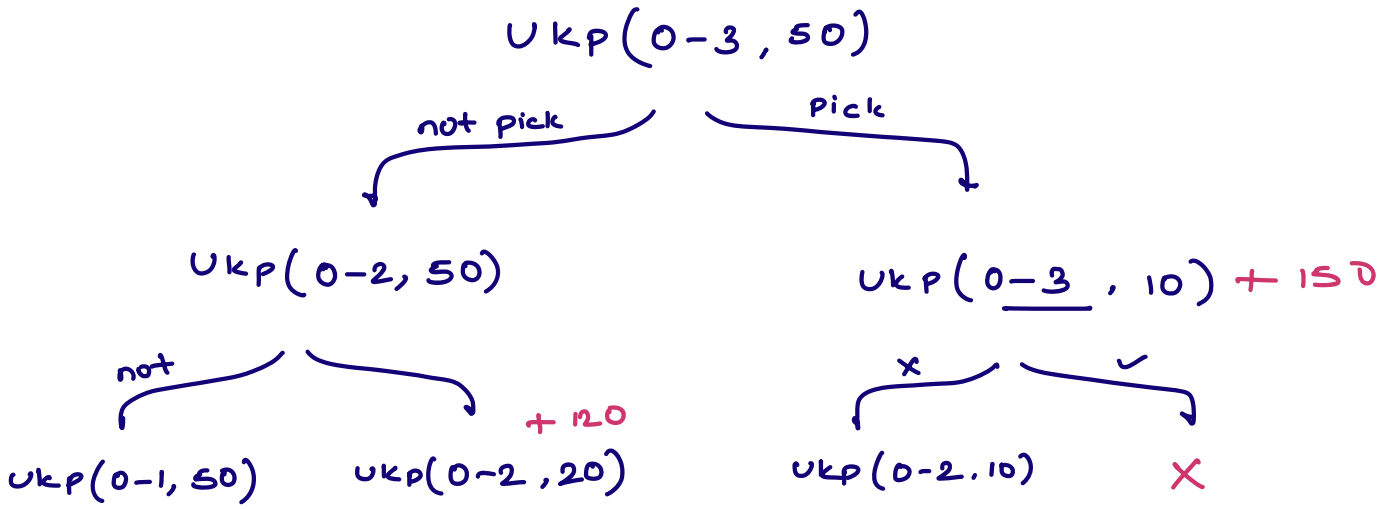
Note 1 :- Every item can be picked infinite no. of times

Note 2 :- We cannot take a part of item

Eg:- N = 4 items K = 50

| Items: | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| Weight[]: | 20 | 10 | 30 | 40 |
| Value[]: | 100 | 60 | 120 | 150 |

Ans = 300



```
for (i=0; i<n; i++)
    for (j=0; j<m; j++){
        if (i==0 || j==0) dp[i][j]=0
        else{
            rej = dp[i-1][j]
            sel = 0
            if ( j ≥ wt[i-1])
                select = val[i-1] + dp[i][j- wt[i-1]];
            dp[i][j]= max( rej, select);
        }
    }
}
return dp[N][K];
```