# 0/1 Knapsack

↳ Dynamic Programming

| Objects | 1 | 2 | 3 |
|---|---|---|---|
| Profit | 10 | 20 | 30 |
| Weight | 1 | 1 | 1 |
| $x_i$ | 0/1 | 0/1 | 0/1 |

(Binary)

$m = 2$

Constraint $\sum w_i x_i \leq m$

Maximum Profit

$$\max \sum p_i x_i$$

DP

1. choice 
2. Optimization Problem
   ↳ maxima, minima

| | $n_1$ | $n_2$ | $n_3$ |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 |
| 5 | 1 | 0 | 0 |
| 6 | 1 | 0 | 1 |
| 7 | 1 | 1 | 0 |
| 8 | 1 | 1 | 1 |

$2^3 = 8$

① fulfil the constraint

② max Profit

$\rightarrow 2^n$

$O(2^n)$

→ Exponential time Complexity

$O(2^n)$

Recursive Tree

$K(\overset{n}{3}, \overset{m}{2})$

- Exclude → $K(2, 2)$
  - Exclude → $K(1, 2)$
    - Exclude → $K(0, 2)$
    - Include → $K(0, 1)$
  - Include → $K(1, 1)$
    - Ex → $K(0, 1)$
    - In → $K(0, 0)$
- Include → $K(2, 1)$
  - Exclude → $K(1, 1)$
    - Ex → $K(0, 1)$
    - In → $K(0, 0)$
  - Include → $K(1, 0)$

# Recursive

```
K(m, wt, pr, n) {
        Base case
        if(n==0 || m==0){

            return 0;
        }

    Recursive function call

        if(weight[n-1] > m){

          1 Exclude that Object

    return  K(m, wt, pr, n-1);
        }

        else {

return max( K(m, wt, pr, n-1),

        pr[n-1] + K(m-wt(n-1), wt,

                pr, n-1));
```

| | $n_1$ | $n_2$ | $n_3$ |
|---|---|---|---|
| Profit | 60 | 100 | 120 |
| Weight | 10 | 20 ✓ | 30 ✓ |
| | 0 | 1 | 1 |

$m = 50$

$$\boxed{\text{Expected Result} = 220}$$

Dynamic Programming

↳ ① Memoization

$\underbrace{dp\,(n+1)}_{\text{rows}}\,\underbrace{(m+1)}_{\text{columns}} \longrightarrow$ 2D Array

to store all the unique function call values

↓

avoid the

Problem of

Re-computation

time complexity

$O(n * m)$

Space complexity : $O(n * m)$

No Recursion

$8-6=2$

## (2) Tabulation Approach

$m = 8$

| $x_i$ | 0 | 1 | 0 | 1 |
|---|---|---|---|---|
| Objects | $n_1$ | $n_2$ | $n_3$ | $n_4$ |
| Profit | 1 | 2 | 5 | 6 |
| | 0 | 1 | 2 | 3 |
| Weight | 2 | 3 | 4 | 5 |
| | 0 | 1 | 2 | 3 |

w

$m \longrightarrow 2$

| p | w | n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 3 | 2 | 0 | 0 | 1 | 2 | 2 | 3 | 3 | 3 | 3 |
| 5 | 4 | 3 | 0 | 0 | 1 | 2 | 5 | 5 | 6 | 7 | 7 |
| 6 | 5 | 4 | 0 | 0 | 1 | 2 | 5 | 6 | 6 | 7 | 8 |

Max Profit $\Longleftarrow$ dp(n)(m)

```
tabulation(m, weight, profit, n) {

    int[][] dp = new int[n+1)(m+1];

  for(i=0 to <=n) {

        for(w=0 to <=m) {

      if(i==0 || w==0) {

(1) ____       dp(i)(w) = 0;
```

if(weight[i-1] > w) {

2 dp[i][w] = dp[i-1][w];

}

if(weight[i-1] <= w) {

Exclude Include

dp[i][w] = max( dp[i-1][w] , profit[i-1] +

dp[i-1][w - weight[i-1]]);

}

time complexity → n×m

Space complexity → n×m