0/1 Knapsack
(Dynamic Prog.)

* 0/1 knapsack problem -

- Here we have knapsack that has a weight limit 'W'.
- There are items $i_1, i_2, i_3, \ldots i_n$ each having weights $w_1, w_2, w_3, \ldots, w_n$ and some benefit (value or profit) associated with it $v_1, v_2, \ldots v_n$.
- Our goal is to maximize benefit such that total weight ~~of~~ inside knapsack is at most W.

$$\max . \sum v_i$$

$$\text{subject to } \sum w_i \le W.$$

- Since this is 0-1 knapsack problem, so we can either take an entire item or reject it completely. We can't break an item and fill the knap-sack.

Solution :- Brute force -

Try all 2^n possible subset of items - (n) items.

Any solution better than brute force ?

* D & C :-

Recall :- 1) Partition problem into subproblem

2) Solve the subproblem

3) Combine sol^n to solve original one.

If subproblem are not independent i.e. Subproblem share sub subproblem then D & C algo repeatedly solves common subsubproblem.

Thus, it does more work than necessary.

* Greedy :-

Fails for 0-1 knapsack problem.

Ex :- item - 3 ~~4~~ 4 — w_i W = 4.

value - 15 ~~16~~ 16 — v_i

Greedy max v_i/w_i ratio ⇒ 15 for $w_i = 3$.

As it is 0-1, we can't take remaining 1 from $w_i = 4$. Hence P = 15

However, if we Take $w_i = 4$. then p = 16.

* Dynamic

Idea is to compute the solution to the subproblems once and store the solution in a table, so that they can be used later repeatedly.

(We trade space with time).

Developing a DP Algo:-

(1) Decompose problem into smaller problems, and find relation between optimal solution of original problem and solutions of smaller problems.

(2) Express solutions of original problem in terms of optimal solutions for smaller problems.

(3) Compute value of an optimal solution in a bottom-up fashion by using table.

DP for knapsack.

(1) We construct an array $V[0..n, 0..W]$.
For $1 \le i \le n$ and $0 \le w \le W$, entry $V[i, w]$ will store the maximum (combined) profit of any subset of items $\{1, 2, .., i\}$ of (combined) weight at most w.

If we can compute all entries of this array, then $V[n, W]$ will contain maximum profit of items that can fit into knapsack. (i.e. soln to our problem).

(2) Recursively defined optimal soln in terms of solutions to smaller problem.

Initially -

$v[0, w] = 0$ for $0 \le w \le W \Rightarrow$ no item

$v[i, w] = -\infty$ for $w < 0 \Rightarrow$ illegal,

Recursive step.

$v[i, w] = \max\left(v[i-1, w], v_i + v[i-1, w - w_i]\right)$

for $1 \le i \le n$, $0 \le w \le W$.

(3). Bottom-up computation.

Algo.

```
Knapsack (v, w, n, W)
{
  for (w=0 to W)  v[0,w]=0
  for (i=1 to n)
    for (w=0 to W)
      if (w[i] ≤ w)
        v[i,w]= max {v[i-1,w], v[i]+v[i-1,w-w[i]]};
      else.
        v[i,w]= v[i-1, w];
  return v[n,W]
}.
```

Complexity :- Clearly $O(nW)$.

Example :-

item	1	2	3	4
value	100	20	60	40
wt	3	2	4	1

Initially :-

v[i,w]	w=0	1	2	3	4	5
i=0	0	0	0	0	0	0
1	0					
2	0					
3	0					
4	0					

$n=4, \quad |W|=5$

Rule :- if $wt[i] > w$ then

$v[i,w] = v[i-1, w]$.

else if $wt[i] <= w$ then

$v[i,w] = \max \{ v[i-1,w], v_i + v[i-1, w - wt[i]] \}$.

Now we fill row $\underline{i=1}$.

$i=1, w=1$, we fill $v[1,1]$.

$wt[1] = 3$.

$3 > 1$ yes.

$v[i,w] = v[i-1,w]$. or $v[1,1] = v[0,1]$
$= 0$.

Same for $v[1,2]$.

$wt[1] = 3$

$3 > 2$ yes.

$v[i,w] = v[i-1,w]$ or $v[1,2] = v[0,2]$
$= 0$.

for $v[1,3)$

$wt[1] = 3$

$3 > 3$ $\underline{\underline{No}}$

$$v[i,w] = \max(v[i-1,w], \ val[i] + v[i-1, w-wt[i]])$$

$$= \max(v[0,3], \ val[1] + v[0, 3-3])$$

$$= \max(0, \ 100+0)$$

$$= \underline{\underline{100}}$$

In this way we fill all the entries of $\underline{\underline{v}}$

v[i,w]	w=0	1	2	3	4	5
i=0	0	0	0	0	0	0
1	0	0	0	100	100	100
2	0	0	20	100	100	120
3	0	0	20	100	100	120
4	0	40	40	100	140	140

This ($V[n, W]$) gives the max. Profit.

However it does not give the items to be selected.

For this —

$i = n, \quad w = W.$

while i and $w > 0$ do

if $V[i, w] \ != V[i-1, w]$ then

mark i^{th} item.

set $w = w - wt[i]$.

set $i = i - 1$

else

set $i = i - 1$

end if

end while.

---

is $V[4, 5] \ != V[3, 5]$.

is $140 \ != 120 \Rightarrow$ Yes

mark 4th item. $\quad w = w - wt[i]$

$= 5 - 1 = 4$

$i = i - 1 = 3.$

is $V[3, 4] \ != V[2, 4]$ $\Big\} \rightarrow$ $i = i - 1$

$100 \ != 100 \Rightarrow$ No. $\quad = 2$

is $v[1,4] \,!= v[0,4]$

is $100 \,!= 0 \Rightarrow$ Yes.

mark 1st item.

set $w = w - wt[i]$
$= 4 - 3$
$= 1$

$i = i - 1$
$= 0$

Now $i == 0$, we stop.

Items we are putting inside the knapsack are 1st and 4th.

Problem for practice.

$W = 10$

i	1	2	3	4
v_i	10	40	30	50
w_i	5	4	6	3

Ans :- Profit = 90

Items $\Rightarrow$ 2nd, 4th.