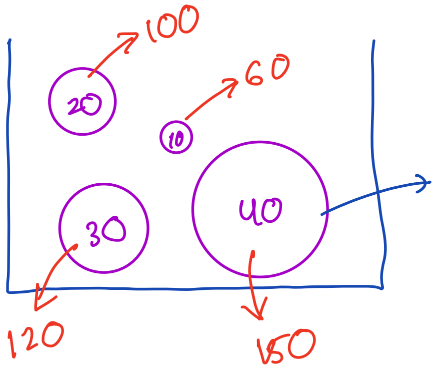
Knapsack (0/1)

Given N items, each with a weight and value.
Find max value by picking items, such that total weight <= K

Note: Every item can be picked only once.
We cannot break the item.

Eg. N = 4 , K = 50

N	1	2	3	4	
W[] =	20	10	30	40	{weights}
V[] =	100	60	120	150	{values}.



40 → 150
10, 40 → 210

20, 30 → 220
20, ~~30, 10~~ →

Take max values

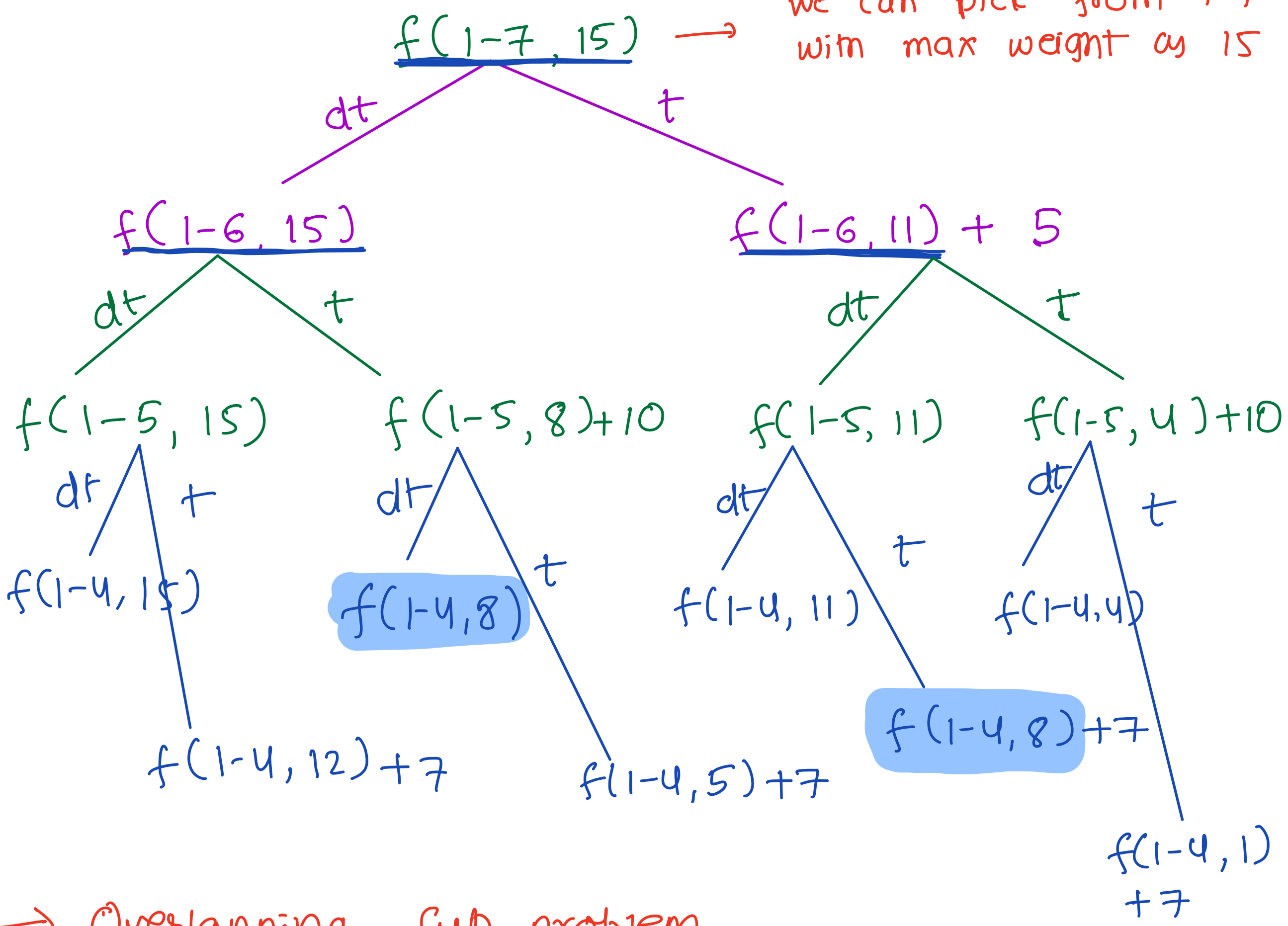
{ 40	30	20	10 }
150	120	100	60

Brute force : Create all the subsets
Check if you dont exceed K } max.

TC: $O(n \cdot 2^n)$

K = 15

items N	1	2	3	4	5	6	7	
w []	4	1	5	4	3	7	4	weights
v []	3	2	8	3	7	10	5	values.



→ we can pick from 1-7 with max weight as 15

→ Overlapping sub problem

→ Optimal substructure

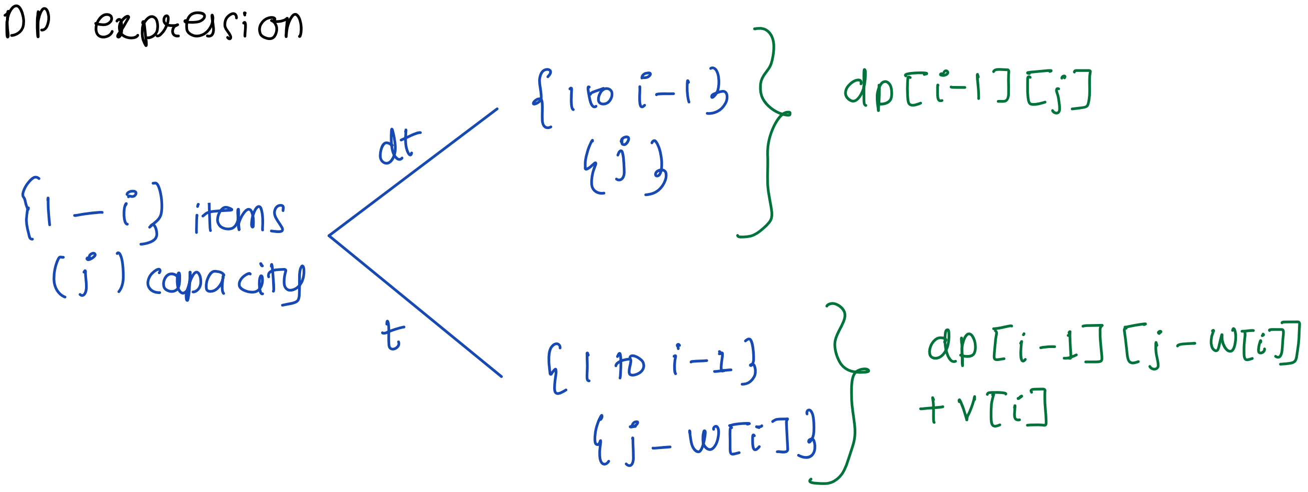
DP state

dp[i][j] = Max value by picking {1 – i} items total weight <= j

items = {1 – i}

Capacity = j

DP expression



$$\boxed{\begin{aligned} dp[i][j] = \max(\ & dp[i-1][j], \\ & dp[i-1][j - w[i]] + v[i]\ ) \end{aligned}}$$

DP Table N items K {capacity}

dp [N+1][K+1]

Base cases

i == 0 from {0 – j} dp[0][j] = 0
no items

j == 0 from {0 – i} dp[i][0] = 0
0 capacity

Pseudo Code

```
int dp [N+1][K+1]
```

```
for ( i=0 ; i <=N ; i++) {
        dp[i][0]  =0
}

for ( j =0 ; j <=K ; j++) {
        dp [0] [j] =0
}

for (i = 1  ; i<= N  ; i++ ) {
    for ( j = 1 ; j<=K  ; j++ ) {
            value   =  V[i-1]
            weight   =  W[i - 1]
            dont    =  dp[i-1][j]
            take    =  0
            if (j >= weight )
                  take    = dp [i-1] [j - weight] +
                                                value

            dp [i][j]  =   max ( take , dont )

    }
}
   return   dp[N] [K]  ;
```

TC : O(N K)

SC : O(NK)

Break   10mins   till   8 : 43

N = 5   K = 8

items	1	2	3	4	5
W[]	3	6	5	2	4
V[]	12	20	15	6	10

weights

items	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	0	12	12	12	12	12	12
2	0	0	0	12	12	12	20	20	20
3	0	0	0	12	12	15	20	20	27
4	0	0	6	12	12	18	20	21	27
5	0	0	6	12	12	18	20	22	27

$(dp[i-1][j] ,$
$dp[i-1][j - w[i]] + V[i])$

$$\begin{tikzcd}
 & dp[i-1][j] \\
dp[i][j] \arrow[ur, "dt"] \arrow[dr, "t"'] & \\
 & dp[i-1][j-W[i]] + V[i]
\end{tikzcd}$$

{ not pick ith item }

{ picked the ith item }.

code >

```
List<Integer> ans ;
 j = K
for (i = N ; i > 0 ; i--) {
|       weight  =  W[i-1]
|       Value   =  V[i-1]
|
|       take  =  dp[i-1][j-weight] + value
```

```
        dont = dp[i-1][j]

        if (take == dp[i][j]) {
            j -= weight;
            ans.add(i);
        }
}
```

dp (2, 6)

dt

dp [1] [6] = 12

t

dp [1] [0] = 20

+ 20

dp [3] [5]

dt

dp [2] [5] = 12

t

dp [2] [5-5] = 15

+ 15

dp [3] [6]

dt

dp [2] [6]

t

dp [2] [1]

+15

dp [3] [8]

dt

dp [2] [8]

tt

dp [2] [3]

+ 15.

dp 5 8

dt

dp 4 8} 27

t

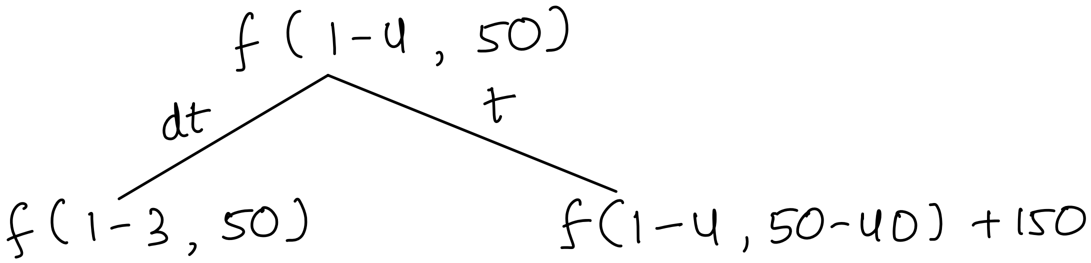
dp 4 4}

+ 10 22

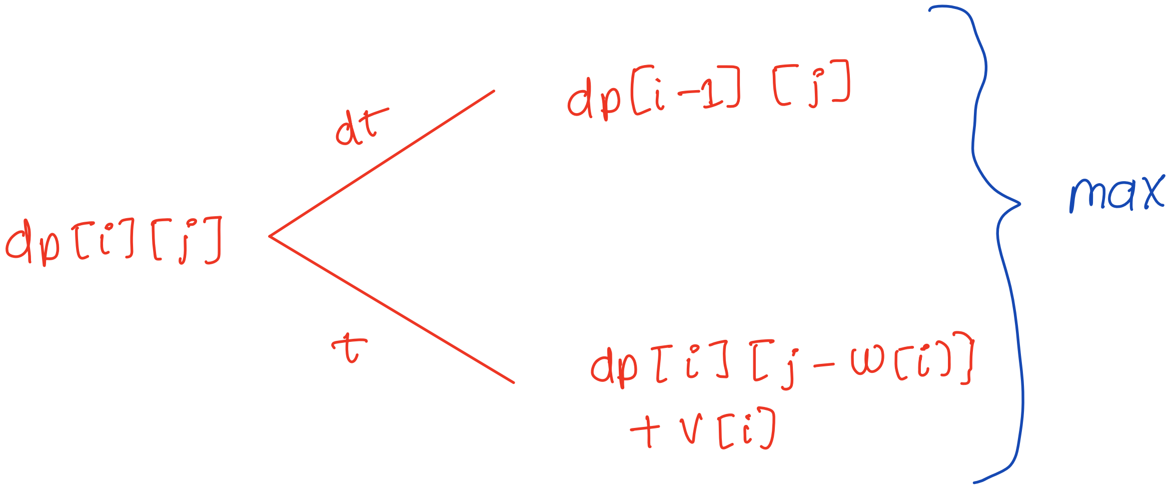
Unbounded knapsack

Q> Exactly the same as above.
A single element can be picked infinitely.

N =	1	2	3	4	k=50
W[]	20	13	10	40	
V[]	100	66	40	150	240



f(1-4, 50)
dt
t
f(1-3, 50)
f(1-4, 50-40) + 150



dp[i][j]
dt
t
dp[i-1][j]
dp[i][j-w[i]] + v[i]
max