DP 3 : Knapsack

Question

You are given array of non-negative integers & target sum. find whether there exists a subset whose sum is equal to target sum.

A : [3 , 34, 4, 12, 5, 2] sum = 9

ans = true

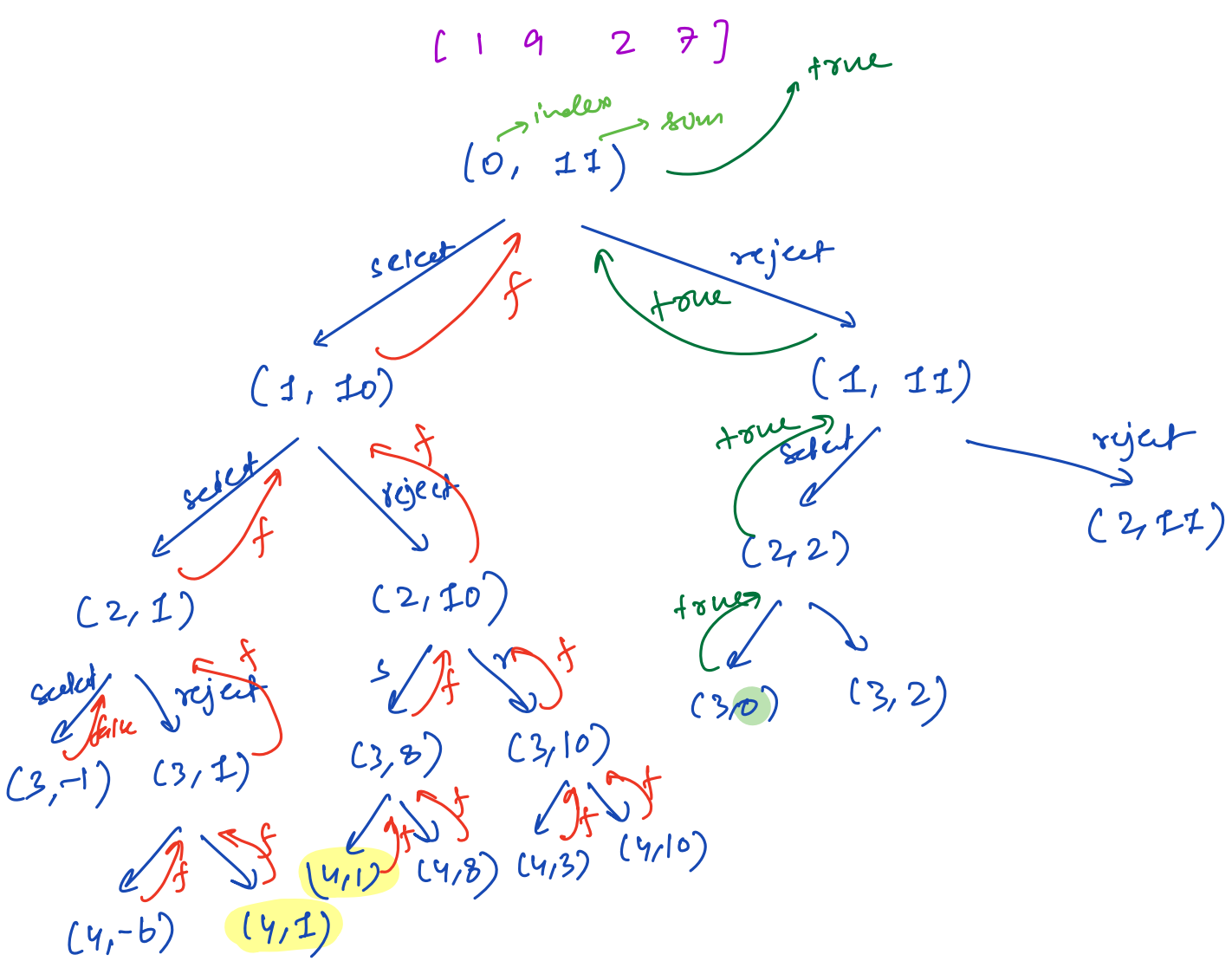
A = [1 2 3 4] sum = 10

ans = true

Bruteforce → Check all subsets and take their sum

[1 9 2 7] sum = 11

for each a[i], we can either select it or reject it.



Code

```
bool isSubsetSum( i , sum) {
    if( sum == 0)   return true;
    if( sum < 0 || i == n)   return false;
    return isSubsetSum( i+1, sum-a[i]) ||
           isSubsetSum( i+1, sum)
}
```

$TC = O(2^N)$

$SC = O(N)$

Apply DP

```
int dp[n][sum+1];
if dp[i][j] = -1;

bool isSubsetSum( i , sum) {
    if( sum == 0)   return true;
    if( sum < 0 || i == n)   return false;
    if ( dp[i][sum] != -1)   return dp[i][sum];

    dp[i][sum] = isSubsetSum ( i+1, sum-a[i]) ||
                 isSubsetSum ( i+1, sum)

    return dp[i][sum];
}
```

[0...n-1] [0....sum]

TC = O(N * sum)

SC = O(N * sum)

Knapsack

Given N objects with their value (profit / loss) & their weight. A bag is given with capacity W that can be used to carry some objects.

Goal is to maximize / minimize profit / loss within the limited capacity.

Fractional Knapsack (objects can be divided)

Question

Given N cakes with their happiness & cost.
find the max. happiness that can be bought with budget W. (cakes can be divided)

N = 5
W = 40

	0	1	2	3	4
H = [	3	8	10	2	5]
C = [	10	4	20	8	15]

↳ 1 unit

40 - 4 = 36
36 - 20 = 16
16 - 15 = 1
1 - 1 = 0

$H = 8 + 10 + 5 + \frac{3}{10} * 1$

$= 23.3$

	0	1	2	3	4
u = [	3	8	10	2	5]
c = [	10	4	20	8	15]

Greedy

	Parts	happiness per unit			
0	10	3/10 = 0.3	→	1 * 0.3 = 0.3	1 - 1 = 0
1	4	8/4 = 2	→	4 * 2 = 8	40 - 4 = 36
2	20	10/20 = 0.5	→	20 * 0.5 = 10	36 - 20 = 16
3	8	2/8 = 0.25			
4	15	5/15 = 0.33	→	15 * 0.33 = 5	16 - 15 = 1

= 23.3

Solution :

1. Sort the cakes w.r.t u(i)/c(i) in descending order.
2. Select cake one by one till the complete budget is used.

TC = O(NlogN)

SC = O(N)

↳ create new array for sorting

Headphone : $60 rating : 9/10

Book : $20 rating : 7/10

Kitchen gadget : $45 rating : 8/10

Budget : $100

(Headphone, Book) : $80 , 16

(Book, Gadget) : $65 , 15

(Headphone, Gadget) : $105 , 17

0-1 Knapsack (objects can't be divided)

Question

Given N toys with their happiness & cost.

Find the max. total happiness that can be bought with budget W.

1.33 0.5 1.25 1.4

0 1 2 3

N = 4 h = [4 1 5 7] 7 - 5 = 2

W = 7 c = [3 2 4 5] 2 - 2 = 0

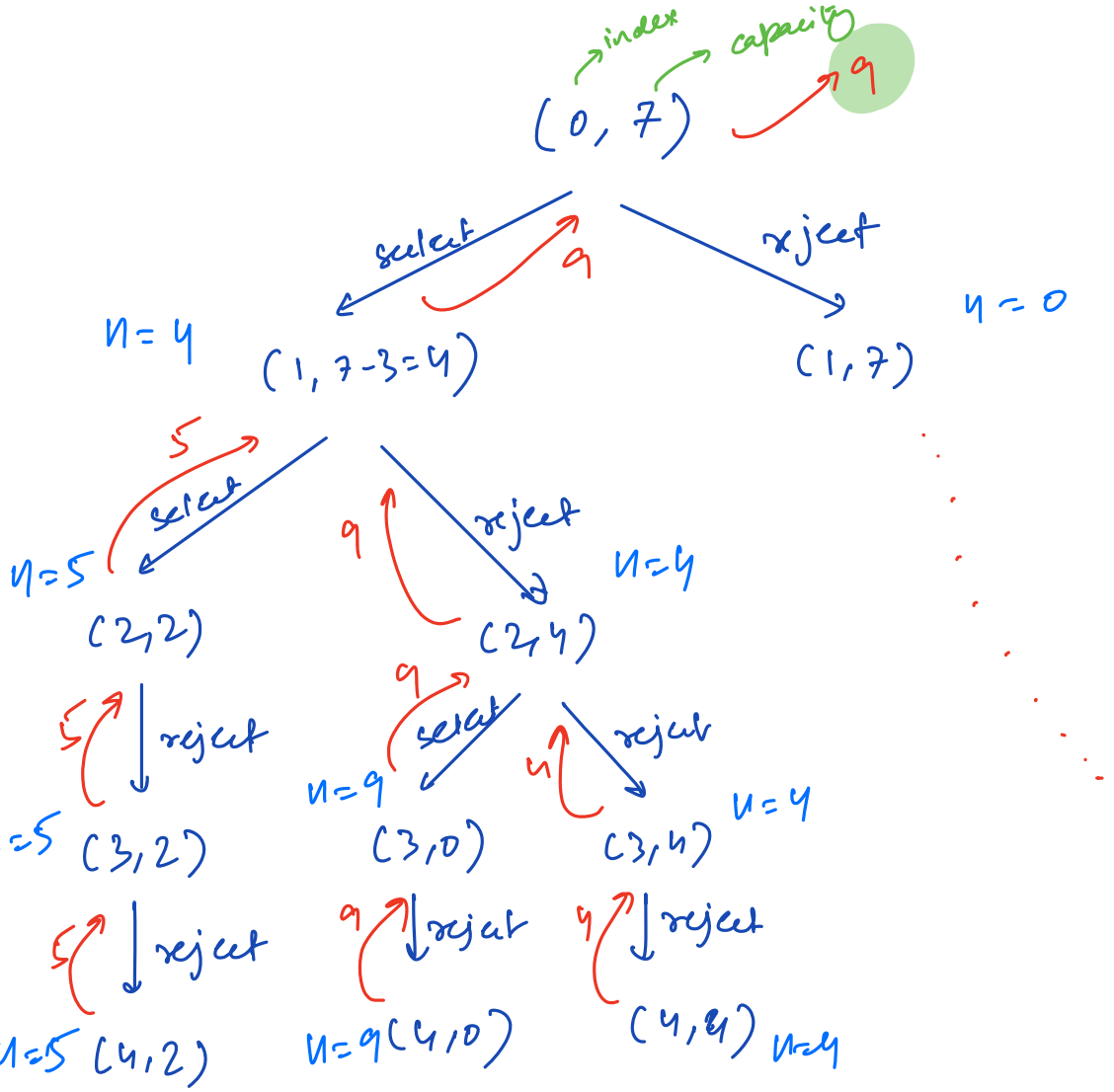
h = 1 + 7 = 8

c = 4 + 3 = 7

h = 5 + 4 = 9

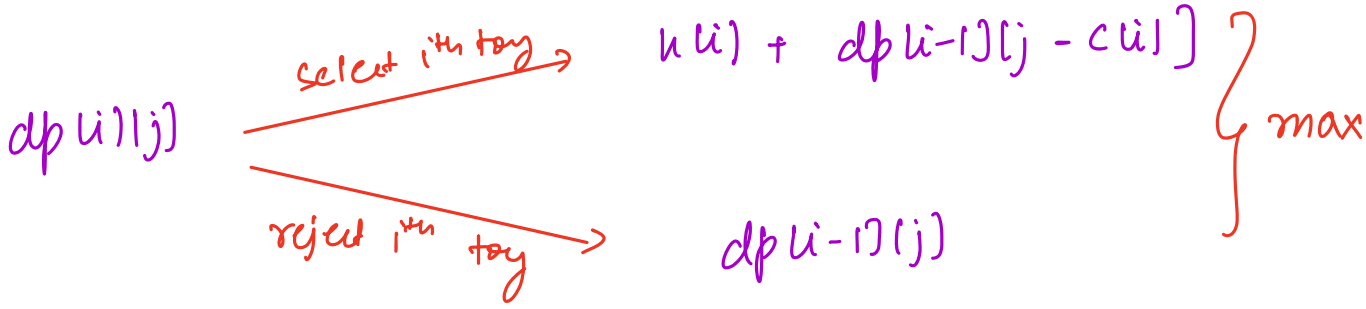
Brute force: if subset of toys, check if total cost <= W, select the max. happiness sum subset.

	0	1	2	3
N = 4	h = [4	1	5	7]
W = 7	c = [3	2	4	5]



dp[i][j] ⟶ max. happiness among toys from index 0 to i with capacity j.

dp[N-1][W] = ?




```
int dp[N][W+1];
fi,j   dp[i][j] = 0;

for(j=1 to W) {          ⟶ i=0
    if(j < c[0])  dp[0][j] = 0;
    else  dp[0][j] = v[0];
}

for(i=1 to n-1) {
    for(j=1 to W) {
        if(j < c[i])   dp[i][j] = dp[i-1][j];
        else {
            dp[i][j] = max(dp[i-1][j], v[i] +
                                dp[i-1][j-c[i]]);
        }
```

```
    }
}
return dp[N-1][W];
```

TC = O(N*W)

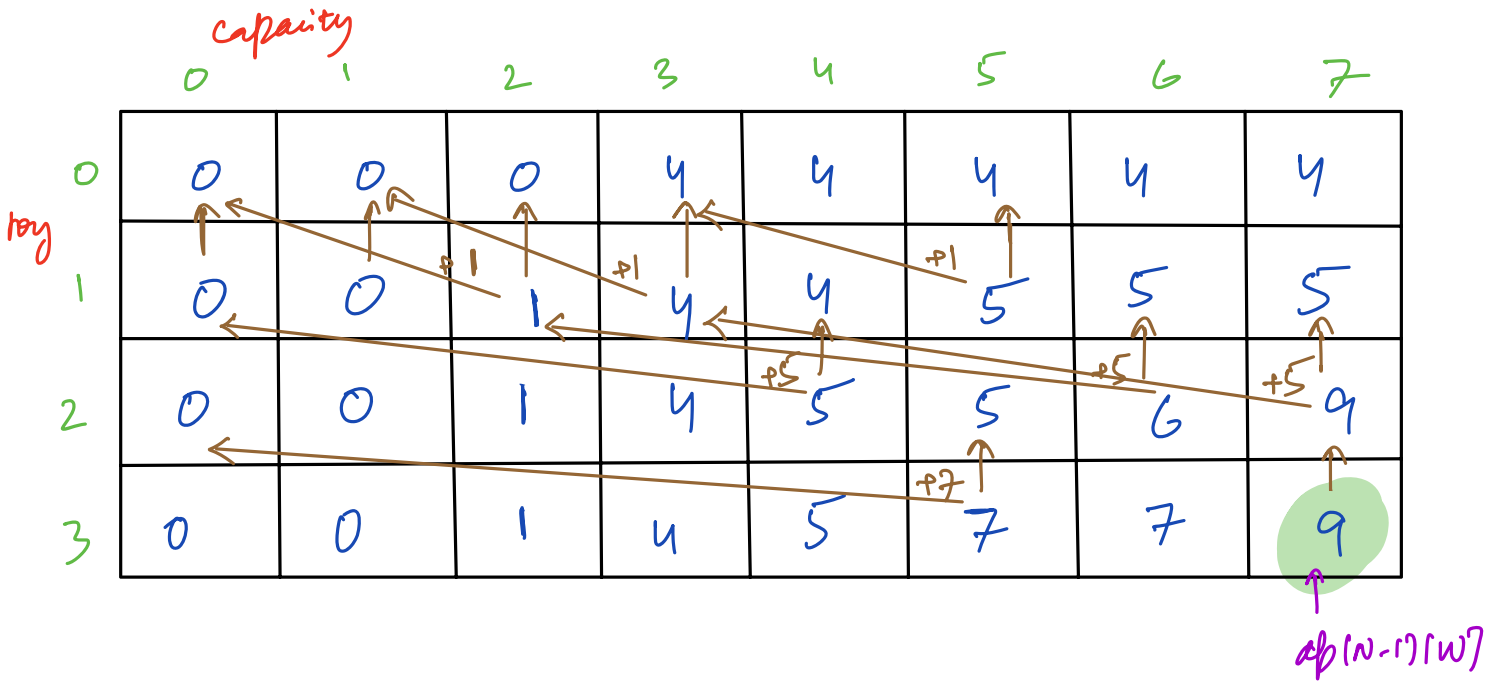
SC = O(N*W)

↓

O(W) by using only 2 rows

N = 4 val = [4 1 5 7]

W = 7 c = [3 2 4 5]



Unbounded Knapsack (objects can't be divided)

(Same object can be selected multiple times)

Question

Given N toys with their happiness & cost.

Find the max. total happiness that can be bought with budget W. (toys can be selected multiple times)

N = 3

W = 8

h = [2 3 5] (0 1 2)

c = [3 4 7]

select 2 times

ans = 6

N = 2

W = 100

h = [1 30]

c = [1 50]

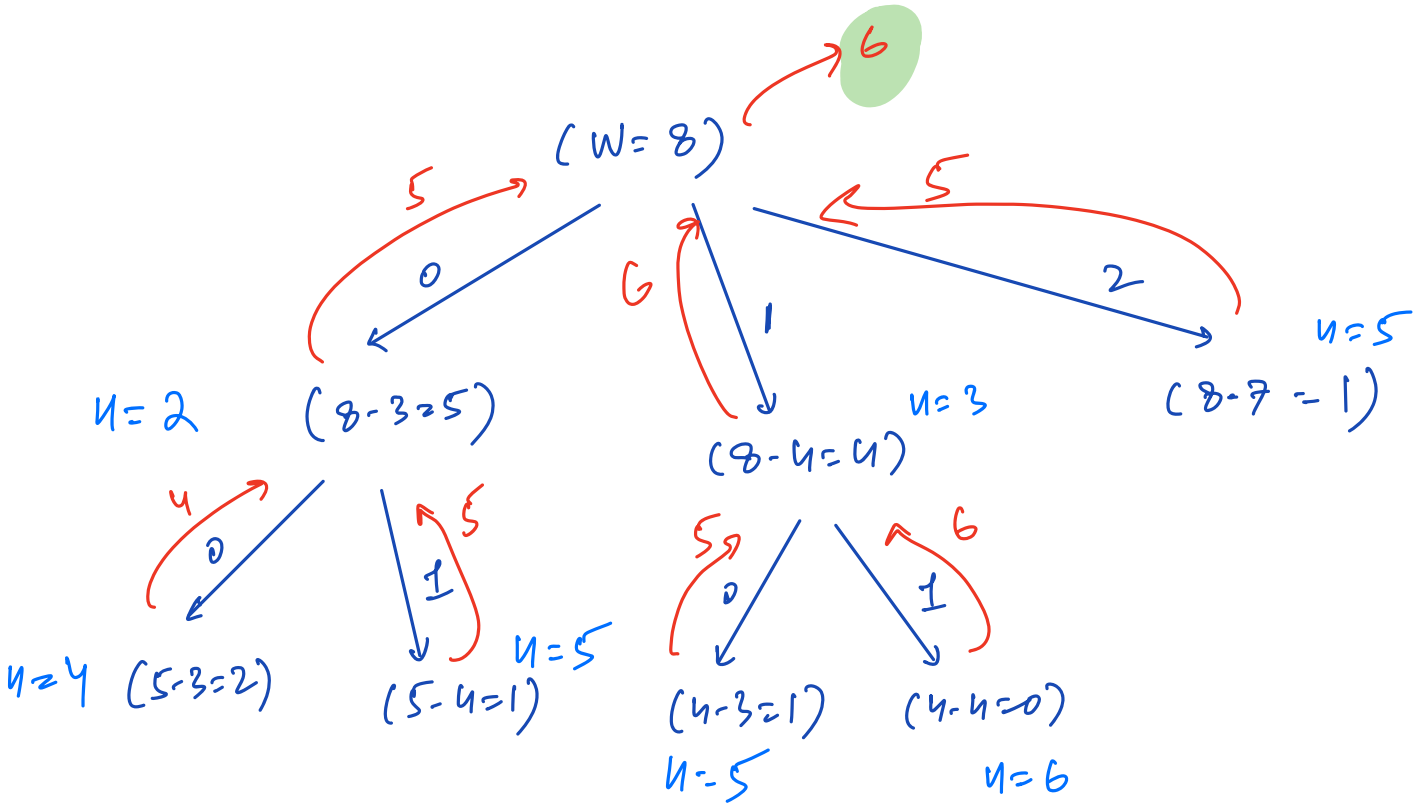
1 — 100 times

30 — 2 times = 60

= 100

$N = 3$ $\quad h = [\underset{0}{2} \quad \underset{1}{3} \quad \underset{2}{5}]$

$W = 8$ $\quad c = [3 \quad 4 \quad 7]$



dp[i] ⟶ max happiness with capacity i

dp[W] = ?

$$dp[i] = \max_{j=0 \text{ to } n-1} \left(h[j] + dp[i - c[j]] \right)$$

code

```
int dp[W+1];

for(i=0, dp[i]=0

for(i=1 to W){

    for(j=0 to N-1){

        if(i >= c[j]){

            dp[i] = max(dp[i], h[j] + dp[i-c[j]]);

        }

    }

}

return dp[W];
```

TC = O(W*N)

SC = O(W)

N = 3

W = 8

h = [2 (0) 3 (1) 5 (2)]

c = [3 4 7]