KNAPSACK PROBLEM

Swarup Kr Ghosh

What is Knapsack?

Item#	Weight (wi)	Value (vi)	
01	1	8	
02	3	6	
03	7	10	
04	5	5	

Number of item = 4
Max weight (capacity of bag)
W =20

Max Σ vi subject to
Σ wi $\leq$ W

Brute Force Approach:

Item#	Weight(Wi)	Value (Vi)	
I1	2	3	
I2	3	4	
I3	4	5	
I4	5	8	
I5	9	10	

Max Weight W= 20

Solution: C1: <I1, I2, I3, I4> <2+3+4+5=14> <3+4+5+8=20>
C2: <I1, I2, I3, I5> <2+3+4+9=18> <3+4+5+10=22>
C3: <I2, I3, I4, I5> <3+4+5+9=21> < reject>
C4: <I3, I4, I5> <4+5+9=18> < 5+8+10=23> ➔ Optimal solution

Time complexity: O(2^n)

Two Approaches:

1. 0-1 Knapsack problem
   - DP approach
2. Fractional Knapsack problem
   - Greedy approach

How to solve the problem?

Criteria-1: Ordering (descending) the item according to the values and start filling taking the maximum value. ➔ 0-1 Knapsack

Criteria-2: Ordering (ascending) the weight and taking the item having minimum weight. ➔ 0-1 Knapsack

Criteria-3: Calculate value per unit weight (Pi=Vi/Wi) and ordering (descending) of Pi and start filling. ➔ Fractional Knapsack

<u>0-1_Knapsack:</u>

Item#	Weight(Wi)	Value (Vi)	
I1	5	30	
I2	10	20	
I3	20	100	
I4	30	90	
I5	40	160	

Max Weight W= 60

Item#	Weight(Wi)	Value (Vi)	vi
I5	40	30	160
I3	20	20	100
I4	30	100	90
I1	5	90	30
I2	10	160	20

< I5,I3> <40+20=60> <160+100=260> ➔ OPTIMAL

Item#	Weight(Wi)	Value (Vi)	
I1	5	30	
I2	10	20	
I3	20	100	
I4	30	90	
I5	40	160	

<I1, I2, I3> <5+10+20=35> <30+20+100=150>

Algorithm: 0-1 Knapsack

Recursive Formula:

B[k, w]= B[k-1, w] **if wk> W**
Max (B[k-1, w], B[k-1, W-wk]+bk) **otherwise**

Procedure: Knapsack01(I, w, v)

```
for w=0 to W
        B[0,w]← 0
for i=0 to n
        B[i,0]← 0
for i=0 to n
        for w=0 to W
                if (w_i ≤ W)
                        if (bi+ B[i-1, W-wi] > B[i-1, w])
                                B[i, w]← bi+ B[i-1, w]
                        else
                                B[i, w]←  B[i-1, W-wi]
                        endif
                else
                        B[i, w]←  B[i-1, w]
                endif
        end for
end for
End procedure
```

Time Complexity: O (n.W)

Fractional Knapsack:

Item#	Weight(Wi)	Value (Vi)	Pi=Vi/Wi
I1	5	30	6.0
I2	10	20	2.0
I3	20	100	5.0
I4	30	90	3.0
I5	40	160	4.0

Max Weight W= 60

Item#	Weight(Wi)	Value (Vi)	Pi=Vi/Wi
I1	5	30	6.0
I3	20	100	5.0
I5	40	160	4.0
I4	30	90	3.0
I2	10	20	2.0

<I1,I3, I5> <5+20(=25)+35/40> <30+100+ 140(35*4.0)=270> GLOBAL OPTIMAL

Algorithm: Fractional Knapsack

Procedure: FracKnapsack(I, w, v)

Step 1: S← {φ} //item to be filled in knapsack

V $\leftarrow$ 0 // value of the selected item

Step 2: for i $\leftarrow$ 1 to n

if ($w_i \leq W$)

S $\leftarrow$ Ii

V $\leftarrow$ V+vi

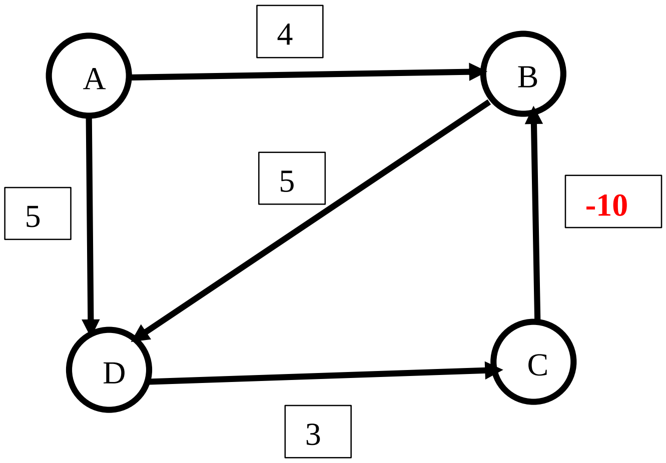
W $\leftarrow$ W-wi

Step 3: return (S, v)

End procedure

Time Complexity: O (n)

Drawback:



Iteration	A	B	C	D
0	0	INF	INF	INF
1	0	4	INF	5
2	0	-2	8	5
3	0	-2	8	3
	0	-4	6	3
	0			

B-D-C: total weigh= -2