

# Analysis and Design of Algorithms

## Session -30

## 7.5 0/1 Knapsack Problem

### Problem statement :

The 0/1 Knapsack problem states that - There are 'n' objects given and capacity of Knapsack is 'm'. Then select some objects to fill The Knapsack in such a way that it should not exceed the capacity of Knapsack and maximum profit can be earned. The Knapsack problem is a maximization problem. That means we will always seek for maximum  $P_i x_i$  (where  $P_i$  represents profit of object  $x_i$ ). We can also get  $\sum P_i x_i$  maximum iff  $-\sum P_i x_i$  is minimum.

$$\text{Minimize profit} - \sum_{i=1}^n P_i x_i$$

$$\text{Subject to} \quad \sum_{i=1}^n W_i x_i$$

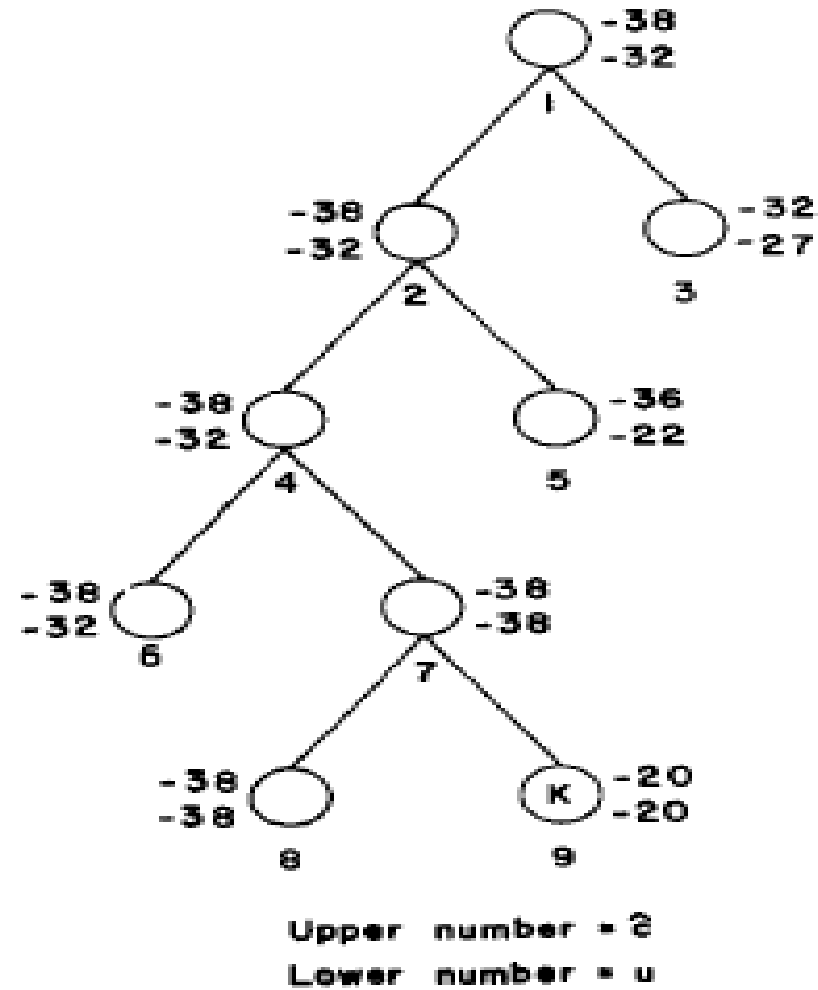
$$\text{Such that } \sum W_i x_i \leq m \text{ and} \\ x_i = 0 \text{ or } 1 \text{ where } 1 \leq i \leq n$$

## Example

Consider the knapsack instance:  $n = 4$ ;

$(p_1, p_2, p_3, p_4) = (10, 10, 12, 18)$ ;

$(w_1, w_2, w_3, w_4) = (2, 4, 6, 9)$  and  $M = 15$ .



**Figure 8.9** LC Branch-and-bound tree for Example 8.2

```
procedure UBOUND ( $p, w, k, M$ )  
  //  $p, w, k$  and  $M$  have the same meaning as in Algorithm 7.11//  
  //  $W(i)$  and  $P(i)$  are respectively the weight and profit of the  $i$ th object//  
  global  $W(1:n), P(1:n)$ ; integer  $i, k, n$   
   $b \leftarrow p; c \leftarrow w$   
  for  $i \leftarrow k + 1$  to  $n$  do  
    if  $c + W(i) \leq M$  then  $c \leftarrow c + W(i); b \leftarrow b + P(i)$  endif  
  repeat  
  return ( $b$ )  
end UBOUND
```

Algorithm 8.5 Function  $u(\cdot)$  for knapsack problem

## 7.5.2 FIFO Branch-and-Bound Solution

The space tree with variable tuple size formulation can be drawn and  $c^{\wedge}(\cdot)$  and  $u(\cdot)$  is computed (We have considered the same Knapsack problem which is discussed in section 7.5.1).

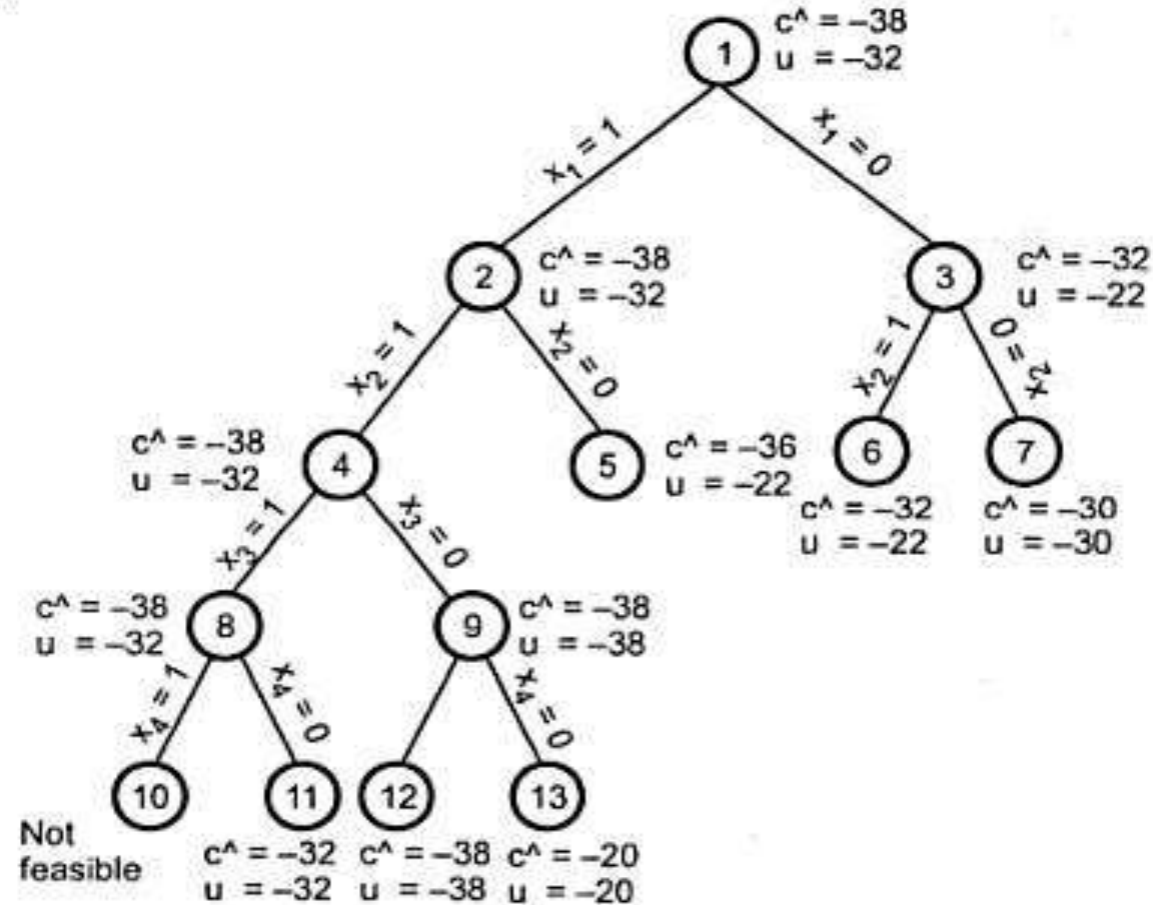


Fig. 7.7 FIFOBB space tree

Initially  $upper = u(1) = -32$ . Then children of node 1 are generated. Node 2 becomes E-node and hence children 4 and 5 are generated. Node 4 and 5 are added in the list of live nodes. Next, node 3 becomes E-node and children 6 and 7 are generated. As  $c^7 > upper$  we will kill node 7. Hence node 6 will be added in the list of live nodes. Node 4 is E-node and children 8 and 9 are generated. The upper is updated and it is now  $upper = u(9) = -38$ . Nodes 8 and 9 are added in the list of live nodes. Node 5 and 6 becomes the next E-node but as  $c^5 > upper$  and  $c^6 > upper$ , kill nodes 5 and 6. Node 8 becomes next E-node and children 10 and 11 are generated. As node 10 is infeasible do not consider it.  $c^{11} > upper$ . Hence kill node 11. Node 9 becomes next E-node and  $upper = -38$ . Children 12 and 13 are generated. But  $c^{13} > upper$ . So kill node 13. Finally node 12 becomes an answer node. Therefore solution is  $x_1 = 1, x_2 = 1, x_3 = 0$  and  $x_4 = 1$ .



## Sample Questions:

1. In Branch and Bound, 0/1 Knapsack problem is a minimization problem. Explain?
2. With suitable example explain LIFOBB
3. Solve 0/1 Knapsack problem using FIFOBB



# THANK YOU