Design And Analysis Of Algorithms

BackTracking

Plan Of The Talk

- Introduction to Backtracking
- Knapsack Problem
- Eight Queen Problem

Introduction To Backtracking

- Backtracking is a general algorithm for finding all solutions to some computational problem, that incrementally builds candidates to the solutions and give up each partial candidate (backtracks) as soon as it determines that candidate cannot possibly be completed to a valid solutions.

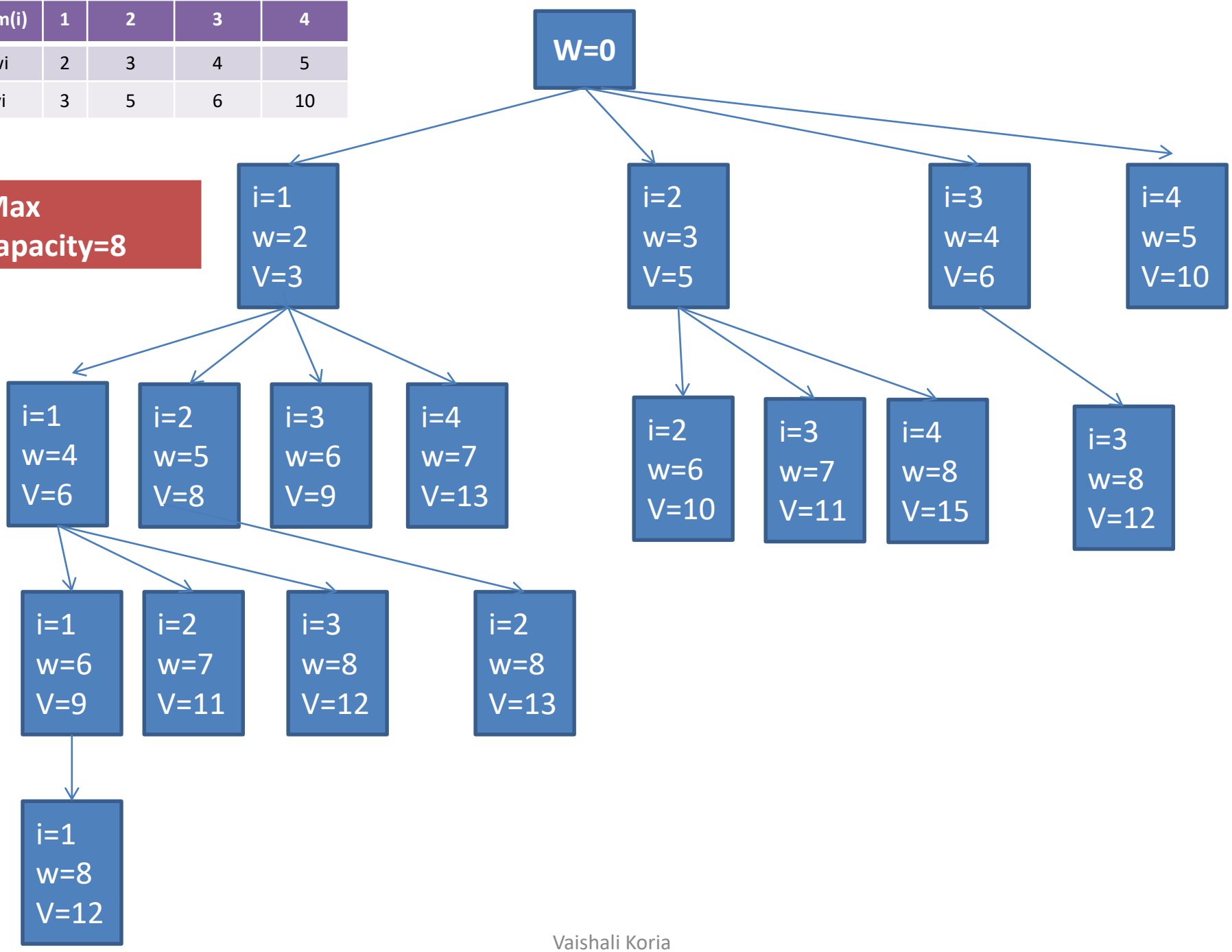
Knapsack

- Example:

Item(i)	1	2	3	4
wi	2	3	4	5
vi	3	5	6	10

Item(i)	1	2	3	4
wi	2	3	4	5
vi	3	5	6	10

Max capacity=8



Eight queens problem

- What is Eight queens problem?
  - Problem of placing 8 queens on 8*8 chess board so that no queens attack each other.
  - No two queens should share same row, column or diagonal.

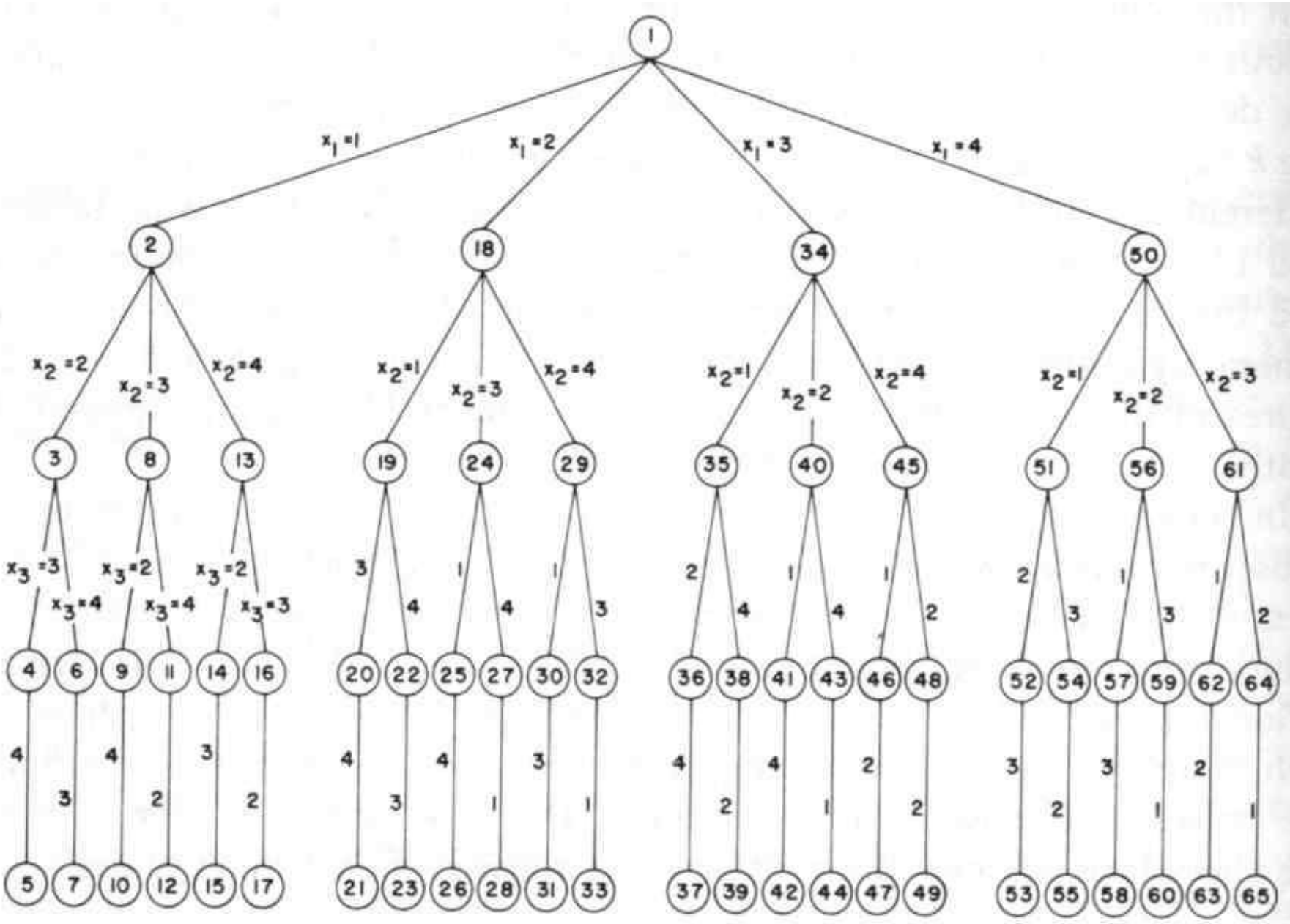
Concept

- Each call places a queen in specific column.
- State of the board from previous placement is known.
- Current Step backtracking:
  - If placement in the column doesn’t give you the solution, then move to the next row in the same column.
- Previous Step backtracking:
  - If all rows in a column have been checked, call terminates and backtracks to the previous column.

Animation for 4 Queens Problem

- http://www.slideshare.net/Tech_MX/8-queens-problem-using-back-tracking

- How to draw the solution space for 4 queen problem?


1
x1 =1
x1 =2
x1 =3
x1 =4
2
18
34
50
x2 =2
x2 =3
x2 =4
x2=1
x2=3
x2=4
x2=1
x2=2
x2=4
x2=1
x2=2
x2=3
3
8
13
19
24
29
35
40
45
51
56
61
x3 =3
x3=2
x3=2
2
1
1
2
1
1
x3=4
x3=4
x3=3
4
6
9
11
14
16
20
22
25
27
30
32
36
38
41
43
46
48
52
54
57
59
62
64
4
4
3
4
3
2
4
4
3
4
3
1
4
4
2
4
2
1
3
3
2
3
2
1
5
7
10
12
15
17
21
23
26
28
31
33
37
39
42
44
47
49
53
55
58
60
63
65

Design And Analysis Of Algorithms

Branch and Bound

Plan Of The Talk

- Introduction to Branch and Bound
- Assignment Problem
- Knapsack Problem

Introduction to Branch and Bound

- **The general idea of B&B is a BFS-like search for the optimal solution, but not all nodes get expanded (i.e., their children generated). Rather, a carefully selected criterion determines which node to expand and when, and another criterion tells the algorithm when an optimal solution has been found.**

- Branch and Bound is a technique that is widely used for speeding up a backtracking algorithm.
- "backtracking with branch and bound".
- We have a recursive algorithm that tries to build a solution part by part, and when it gets into a dead end, then it has either built a solution or it needs to go back (backtrack) and try picking different values for some of the parts.
- We check whether the solution we have built is a valid solution only at the deepest level of recursion –when we have all parts picked out.
- Branch and bound says that sometimes, we can notice that after building only a partial solution there is no need to go any deeper because we are heading into a dead end.

Assignment Problem

- **Input: n jobs, n employees, and an n x n matrix A where A_{ij} be the cost if person i performs job j.**
- **Problem: find a one-to-one matching of the n employees to the n jobs so that the total cost is minimized.**

Example

	T1	T2	T3
P1	5	8	4
P2	3	7	2
P3	4	10	5

If

Persons	Task	Cost	Total Cost is 17
p1	T3	4	
p2	T1	3	
p3	T2	10	

If

Persons	Task	Cost	Total Cost is 16
p1	T2	8	
p2	T1	3	
p3	T3	5	

Optimal assignment for this is:

Persons	Task	Cost	Total Cost is 14
p1	T2	8	
p2	T3	2	
p3	T1	4	

Cost matrix:

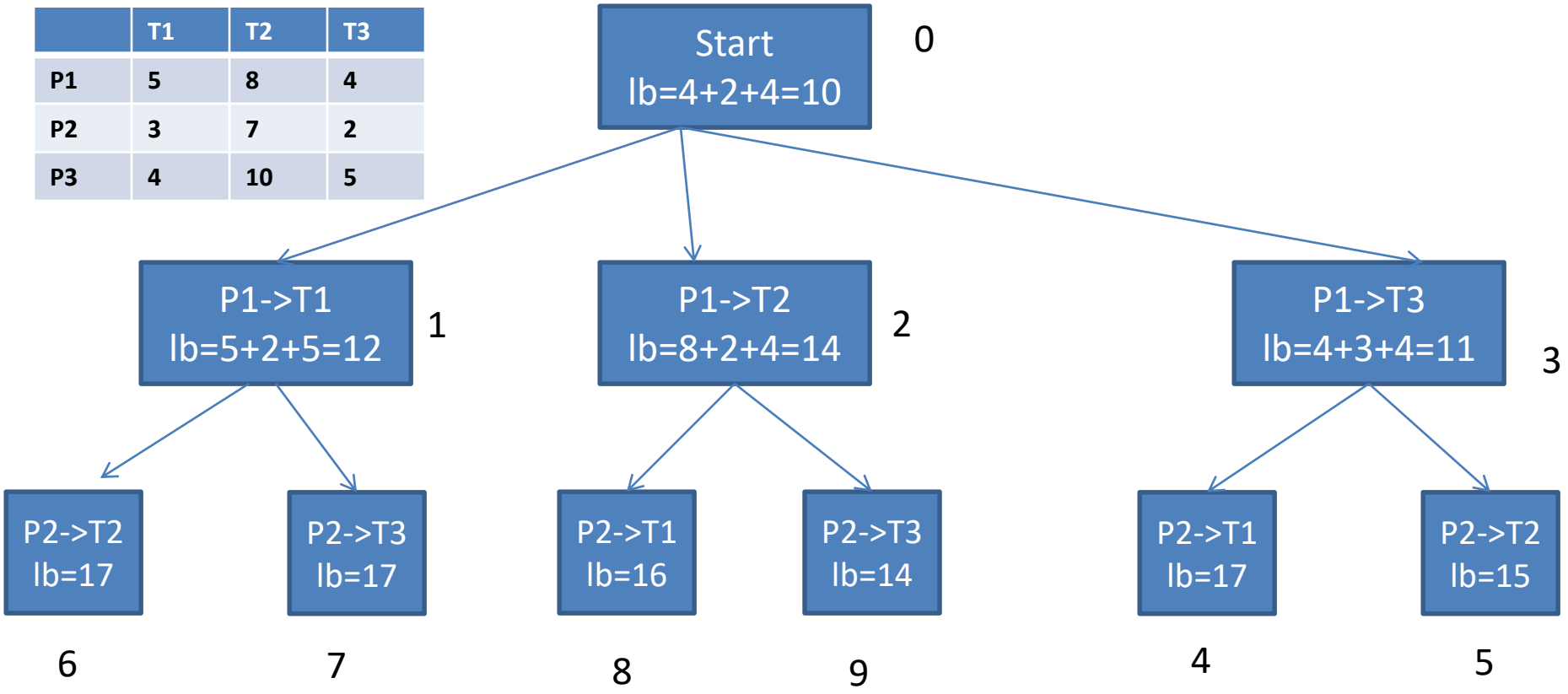
	T1	T2	T3
P1	5	8	4
P2	3	7	2
P3	4	10	5

- Cost of any solution could not be less than the lower bound.
- Lower bound= Sum of the minimum value from each row
- For this example it is lb= 4+2+4=10

Steps:

- Assign T1 to P1, T2 to P1, T3 to P1 and calculate lb for these three possibilities.
- Select the node having minimum lb.
- Give the nodes numbers according to their visit.
- If the lower bounds of the nodes exceed the lb of node labelled 1, explore it and calculate lower bound.
- After examining each leaves in these order, select the assignment with optimal lower bound.

	T1	T2	T3
P1	5	8	4
P2	3	7	2
P3	4	10	5



So the final assignment solution would be

P1->T2 (8)

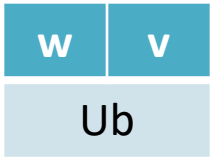
P2->T3 (2)

P3->T1 (4)

Cost is : 14

Knapsack Problem

- Fill the knapsack of capacity W, with a given set of items I1,I2,I3 having weights w1,w2,w3 in such a manner that the total weight of the items should not exceed the knapsack capacity and maximum value can be obtained.
- Items are arranged in descending order of value per weight ratio.
- We have bound that none of these items can have total sum more than knapsack capacity.
- The tree is constructed as binary tree, where left branch signifies the inclusion of the item and right branch signifies exclusion.
- Structure of the node is:



$$Ub = v + (W - w) * (v_{i+1} / w_{i+1})$$

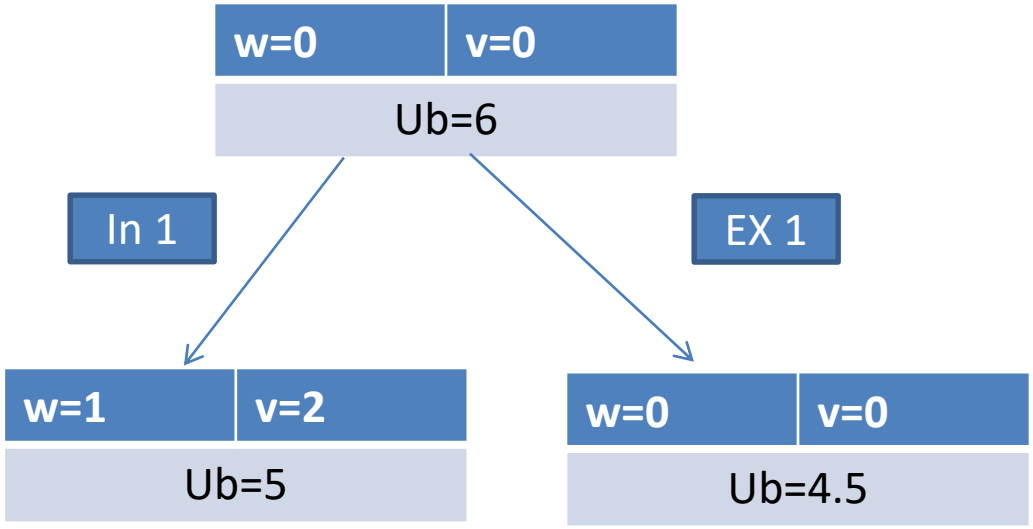
Items	w	v	vi/wi
I1	1	2	2
I2	2	3	1.5
I3	3	4	1.3

W=3

1. Start from the root node.
upper bound is :
v=0, w=0, W=3, v1/w1=2
Ub= 0+(3-0)*2
node is now:

w=0	v=0
Ub=6	

2. Include item 1 indicated by the left branch, and exclude item 1,which is indicated by right branch.

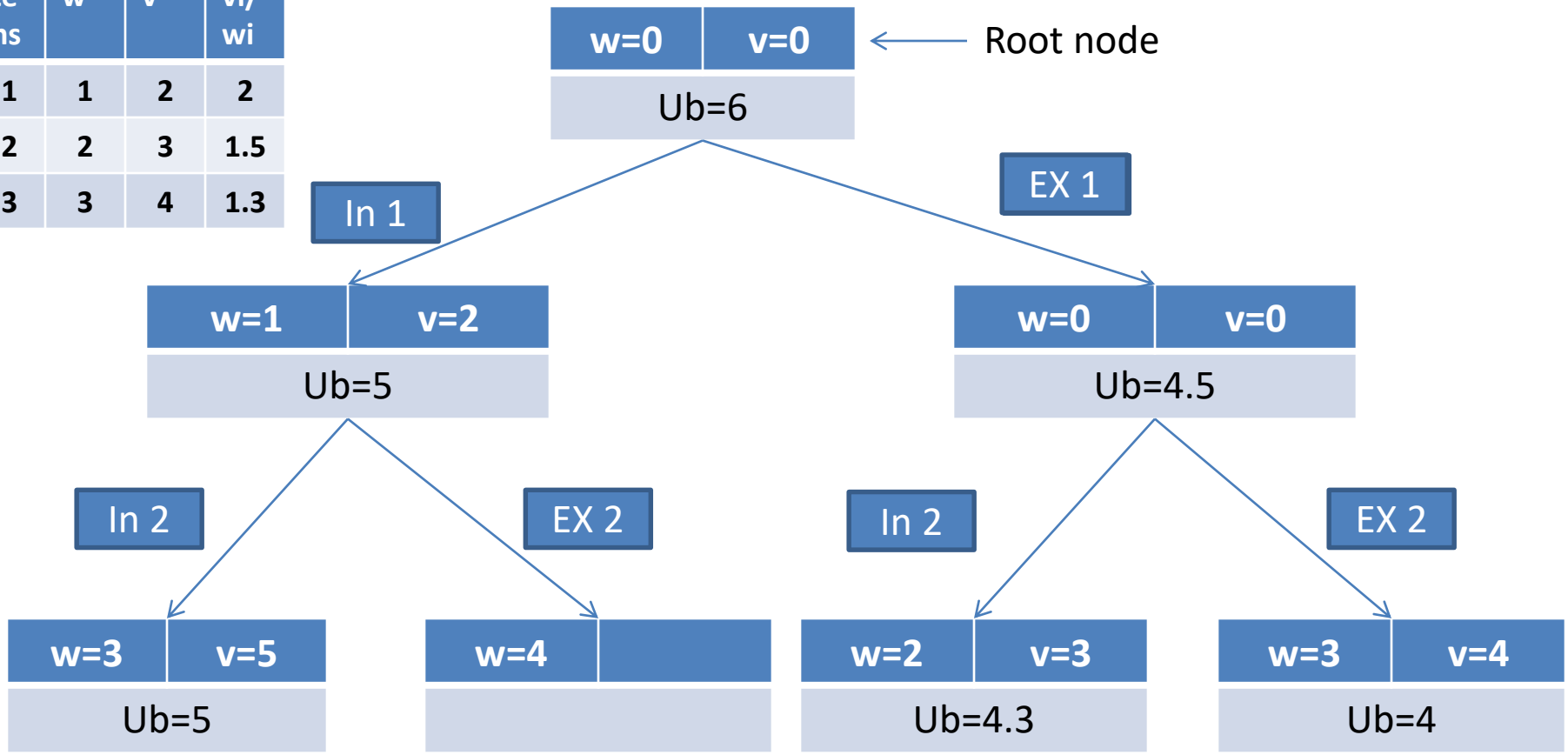


Ub= 2+(3-1)*1.5
=2+(2*1.5)
=2+3=5

EX 1 means Exclude item 1, consider the next item as item 2.

Ub= 0+(3-0)*1.5
=0+(3*1.5)
=4.5

Items	w	v	vi/wi
I1	1	2	2
I2	2	3	1.5
I3	3	4	1.3



- At every level, compute the upper bound, and explore the node while selecting the item.
- Finally, the node with maximum upper bound is selected as an optimal solution.
- In this example, item 1 and item 2 gives optimum solution.