Design and Analysis of Algorithms

L46: FIFO Branch and Bound

0-1 Knapsack Problem

Dr. Ram P Rustagi
Sem IV (2019-H1)
Dept of CSE, KSIT/KSSEM
rprustagi@ksit.edu.in

Resources

- Text book 2: Horowitz
  - Sec 8.2
- Text book 1: Levitin
  - Sec 12.1, **12.2**
- R1: Introduction to Algorithms
  - Cormen et al.
- Youtube link for lecture recording
  - https://www.youtube.com/watch?v=Ns8018VzkrI

$0-1$ Knapsack Problem

- Knapsack problem:
  - maximize $\Sigma_{1 \le i \le n} v_i x_i$,
  - subject to $\Sigma_{1 \le i \le n} w_i x_i \le m$
  - x_i is 0 or 1, and $1 \le i \le n$
  - Note: All the weights w_i's and knapsack capacity m are integers, but values v_i's can be real numbers.
- $0-1$ knapsack is a maximization problem
  - Branch and Bound solves minimization problem.
  - So convert knapsack to minimization problem

BB Search: State Space Tree

- Three possible implementation of search space
  - Depends upon how the list `L` of live nodes is implemented
- **L is Queue** i.e. **FIFO** (First In First Out)
  - E-nodes are removed in the order they are added
  - Also called BFS (Breadth First search)
- L is Stack i.e. LIFO (Last in First Out)
  - E-nodes are removed in the reverse order it is added
  - Also called D-search (Depth First search)
- L is Heap (can be min or max heap)
  - E-nodes are removed as min (or max) value
  - Called Least Cost (LC) Search

$0-1$ Knapsack Problem

- Convert knapsack maximization to minimization

  minimize $-\Sigma_{1 \le i \le n} v_i x_i$, (call it cost)

  – it maximizes $\Sigma_{1 \le i \le n} v_i x_i$ (values)

  subject to $\Sigma_{1 \le i \le n} w_i x_i \le m$ (knapsack constraint)
- State space tree formation
  - Using fixed tuple size, one variable for each weight
    - Each variable has two values 0 or 1
    - Thus, Each node has two children
  - Using variable tuple size, uses the index of weight
    - Can be easily built from fixed tuple size case

$0-1$ Knapsack Implementation

- Define $\hat{c}(x)$: a heuristic value for $c(x)$
  - cost till the first node which doesn't fit the knapsack
    - Thus, include its partial value to max the knapsack
- Define $u(x)$: an upper bound for node x.
  - the cost till the first node which doesn't fit the knapsack, but without including the partial value.
- Thus, two functions follows the constraints for node x
  $\hat{c}(x) \leq c(x) \leq u(x)$
- Maintain single `upper` variable.
  - This indicates the best value i.e. minimum cost solution achieved so far.
- Thus, for any node when $\hat{c}(x) > upper$
  - Discard that path (i.e. kill that node), prune the tree

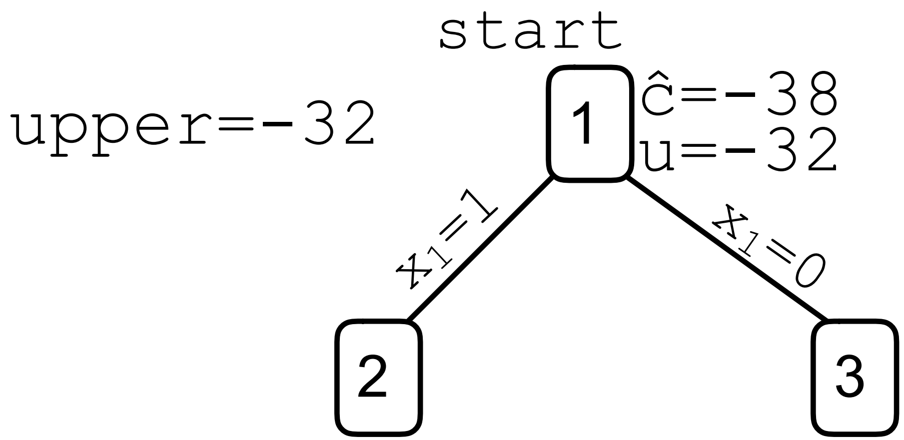
Example: FIFOBB

- Consider knapsack instance with $n=4$, $m=15$, and
  - values $(v_1, v_2, v_3, v_4)=(10, 10, 12, 18)$, and
  - weights $(w_1, w_2, w_3, w_4)=(2, 4, 6, 9)$
  - Fixed implementation implies 4 tuple varaibles
    - x_1, x_2, x_3, x_4 and each can take value 0 or 1.
- Using fixed tuple implementation, trace FIFOBB
- First In First Out (FIFO) approach
  - Among live nodes, choose that node to explore (E-node) which was added first to queue
    - and not the least cost.

FIFOBB: 0-1 Knapsack

	1	2	3	4
v	10	10	12	18
w	2	4	6	9

n=4,m=15



- **start node** 1:

  ĉ(x): w_1, w_2, and w_3 contributes fully, w_4 exceeds knapsack

  ĉ=-(10+10+12+((15-12)/9)*18))=-38

  u(x): w_1, w_2, and w_3 contributes fully, w_4 exceeds knapsack

  u=-(10+10+12+0)=-32]

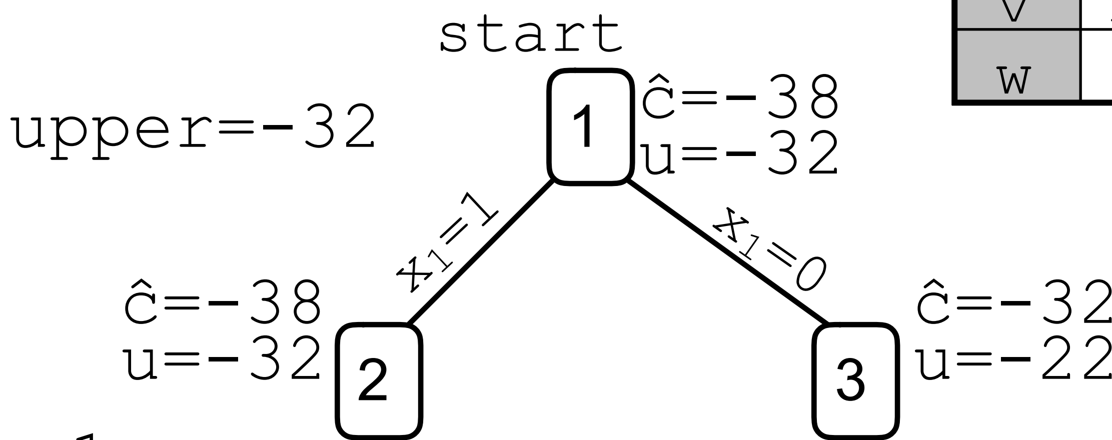
  upper=-32

  - This node is live node (ĉ≤upper) and only node so far,
  - Explore this node, two children
    - x_1=1 (include w_1), x_1=0(exclude w_1)

FIFOBB: 0-1 Knapsack

	1	2	3	4
v	10	10	12	18
w	2	4	6	9

n=4, m=15

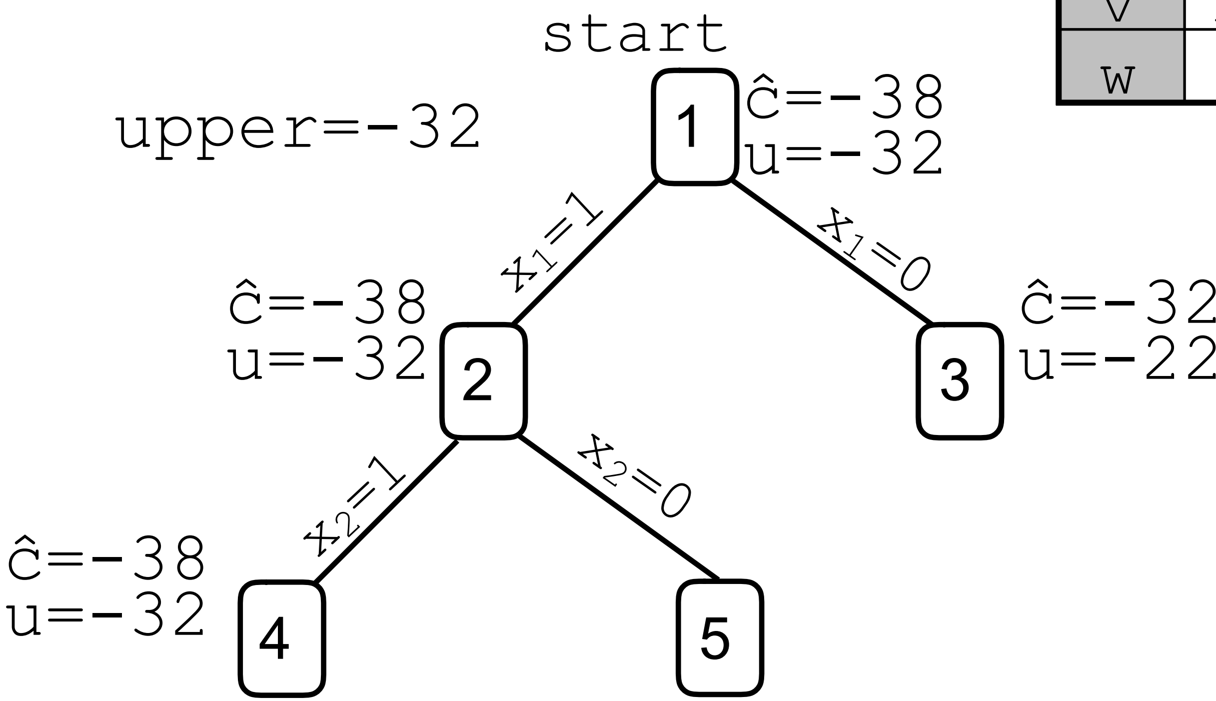


- **node** 2: x_1=1
  - ĉ(x): w_1, w_2, **and** w_3 **contributes fully,** w_4 **exceeds knapsack**
    - ĉ=-((10+10+12+((15-12)/9)*18))=-38
  - u(x): w_1, w_2, **and** w_3 **contributes fully,** w_4 **exceeds knapsack**
    - u=-(10+10+12+0)=-32
    - ĉ(x), u(x), upper **don't change**
- **node** 3: x_1=0 **(partial weight of** w_4 **becomes** 15−10=5**)**
  - ĉ=-(0+10+12+((15-10)/9)*18))=-32
  - u=-(0+10+12+0)=-22
  - upper **remains** −32 **(doesn't change)**
- **Lives nodes** : {2,3}, (ĉ(x) ≤ upper)

FIFOBB: 0-1 Knapsack

	1	2	3	4
v	10	10	12	18
w	2	4	6	9

n=4, m=15



- Using FIFO for {2, 3} live nodes, next E-node is 2.
- Explore node 2.
  - $x_2=1$ (node 4), and $x_2=0$(node 5)
- Node 4:

  $\hat{c}=-(10+10+12+((15-12)/9)*18))=-38$

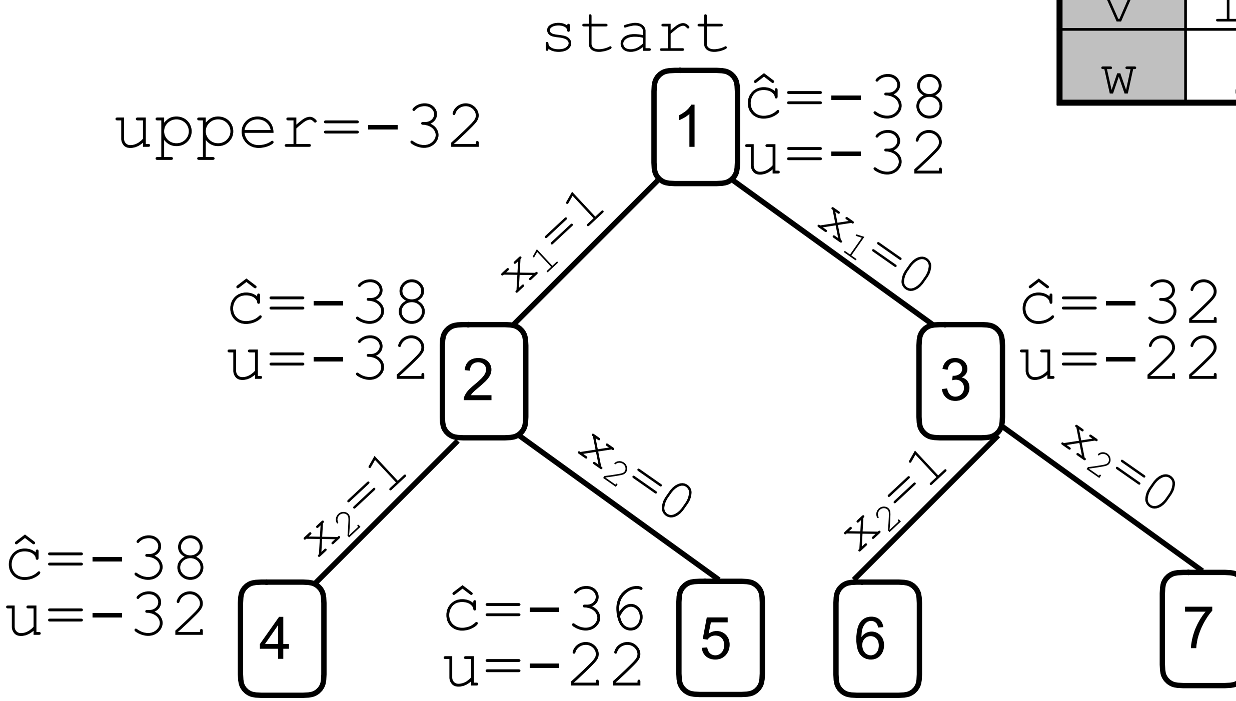
  $u=-(10+10+12+0)=-32$

  upper remains same and doesn't change

FIFOBB: 0-1 Knapsack

	1	2	3	4
v	10	10	12	18
w	2	4	6	9

n=4,m=15

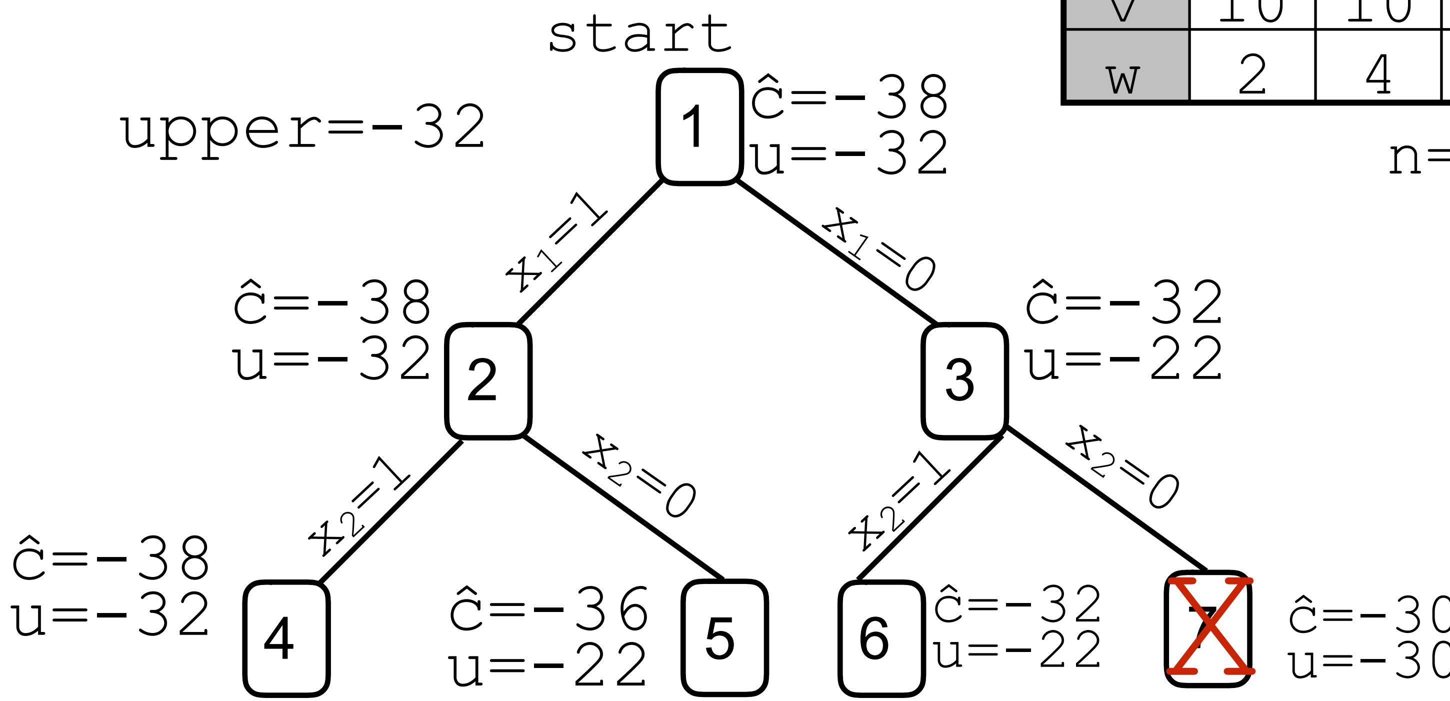


- **Node** 5 (partial weight of w_4 becomes 15−(2+6)=7)
  ĉ=-(10+0+12+((15-8)/9)*18))=-36
  u=-(10+0+12+0)=-22
  upper remains same and doesn't change
- Lives nodes now: {3,4,5}; (c(x)≤upper)
- E-node using FIFO is 3. Explore it
  - x_2=1 (node 6), and x_2=0(node 7)

FIFOBB: 0-1 Knapsack

	1	2	3	4
v	10	10	12	18
w	2	4	6	9

n=4,m=15



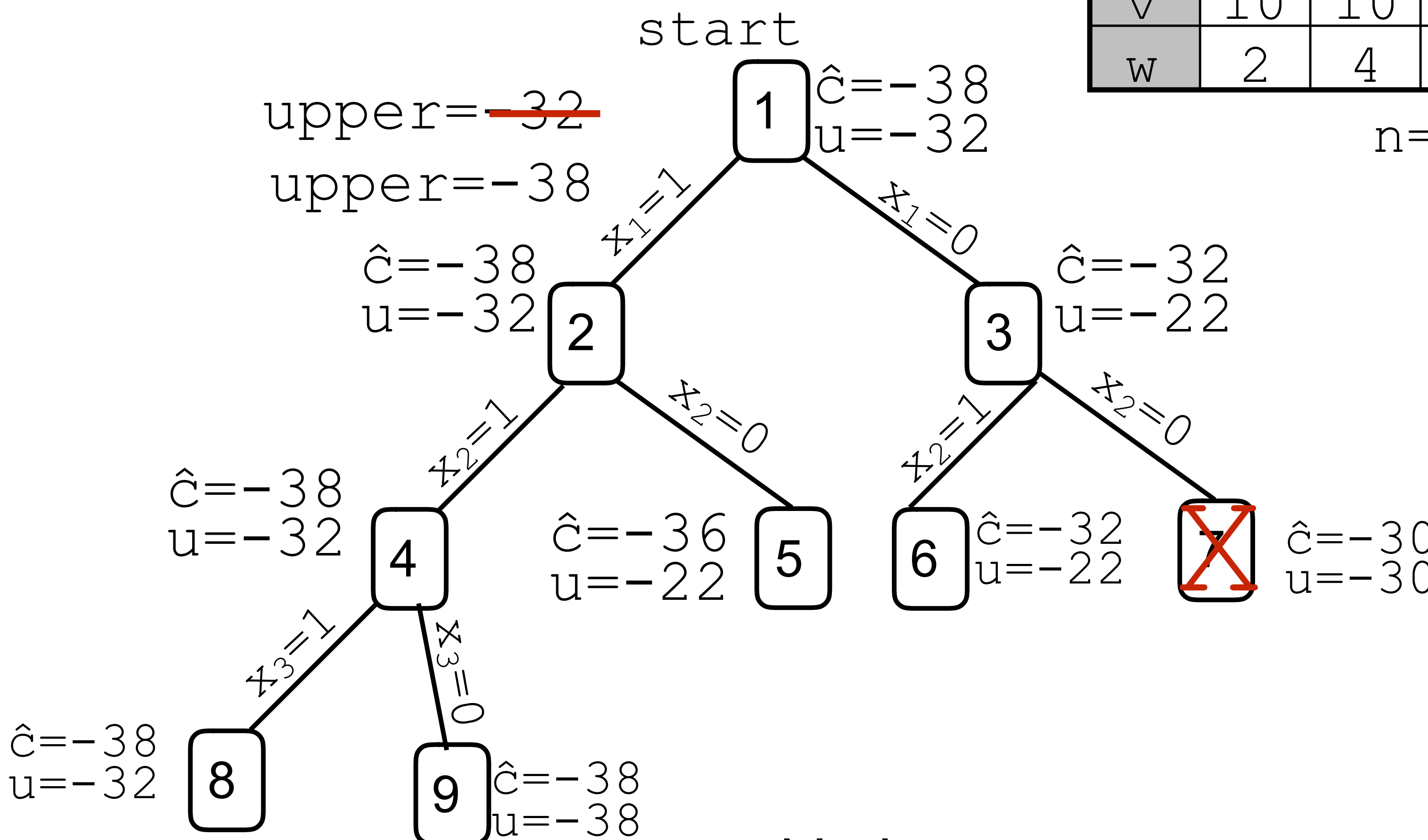
- Node 6 (x_2=1)
  ĉ=-(0+10+12+((15-10)/9)*18))=-32
  u=-(0+10+12+0)=-22
  upper **remains same and doesn't change**
- Node 7 (x_2=0)
  ĉ=-(0+0+12+18))=-30
  u=-(0+0+12+18)=-30
  ĉ(7)>upper, **thus kill this node**

- **Lives nodes now:** {4,5,6}; (c(x)≤upper)
- **Next E-node:** 4

FIFOBB: 0-1 Knapsack

	1	2	3	4
v	10	10	12	18
w	2	4	6	9

n=4,m=15

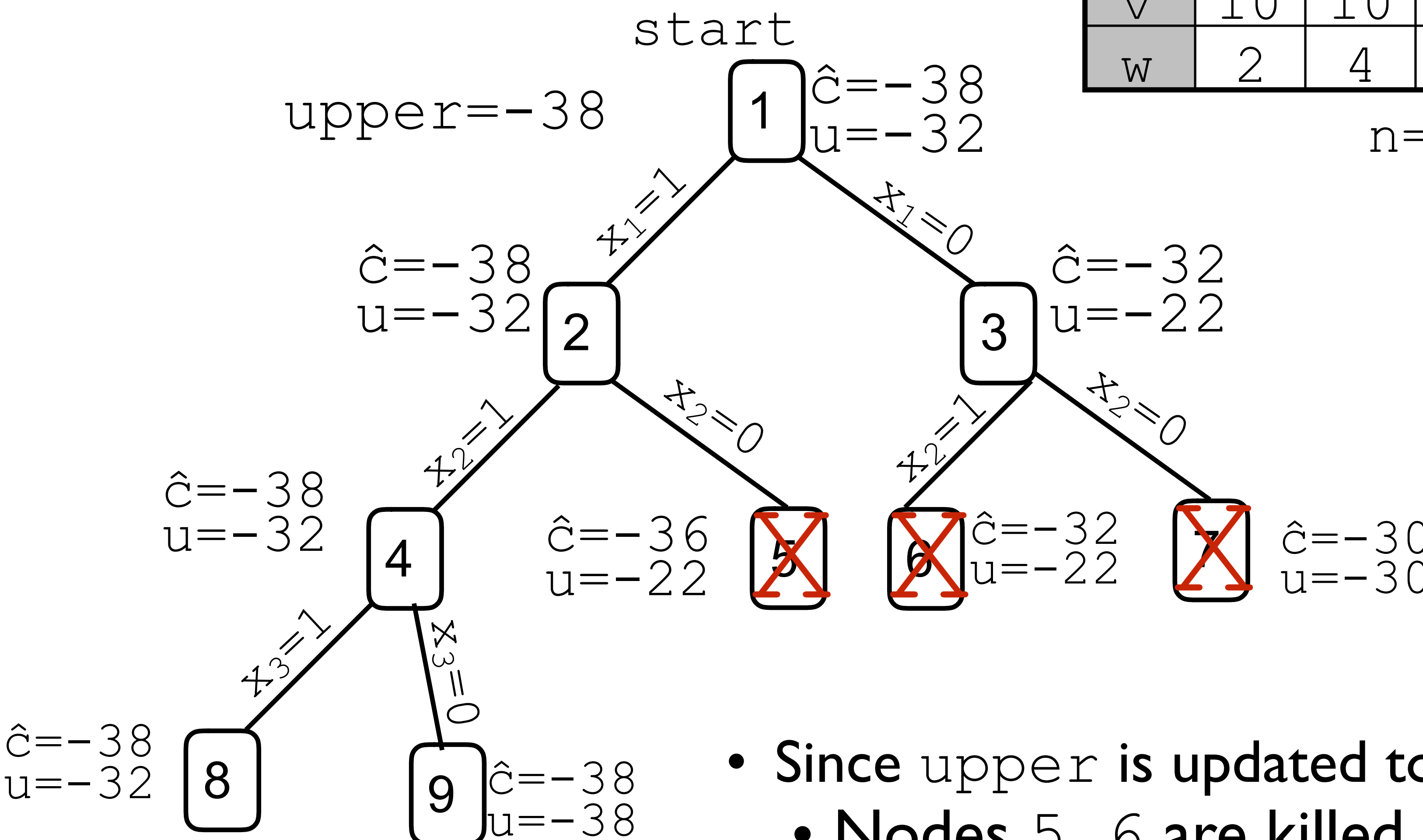


- **Node** 8 $(x_3=1)$
  $\hat{c}=-(10+10+12+((15-12)/9)*18))=-38$
  $u=-(10+10+12+0)=-32$
  upper **remains same**
- **Node** 9 $(x_3=0)$
  $\hat{c}=-(10+10+0+15=-38$
  $u=-(10+10+0+18)=-38$
  upper **is updated to** -38

FIFOBB: 0-1 Knapsack

	1	2	3	4
v	10	10	12	18
w	2	4	6	9

n=4,m=15

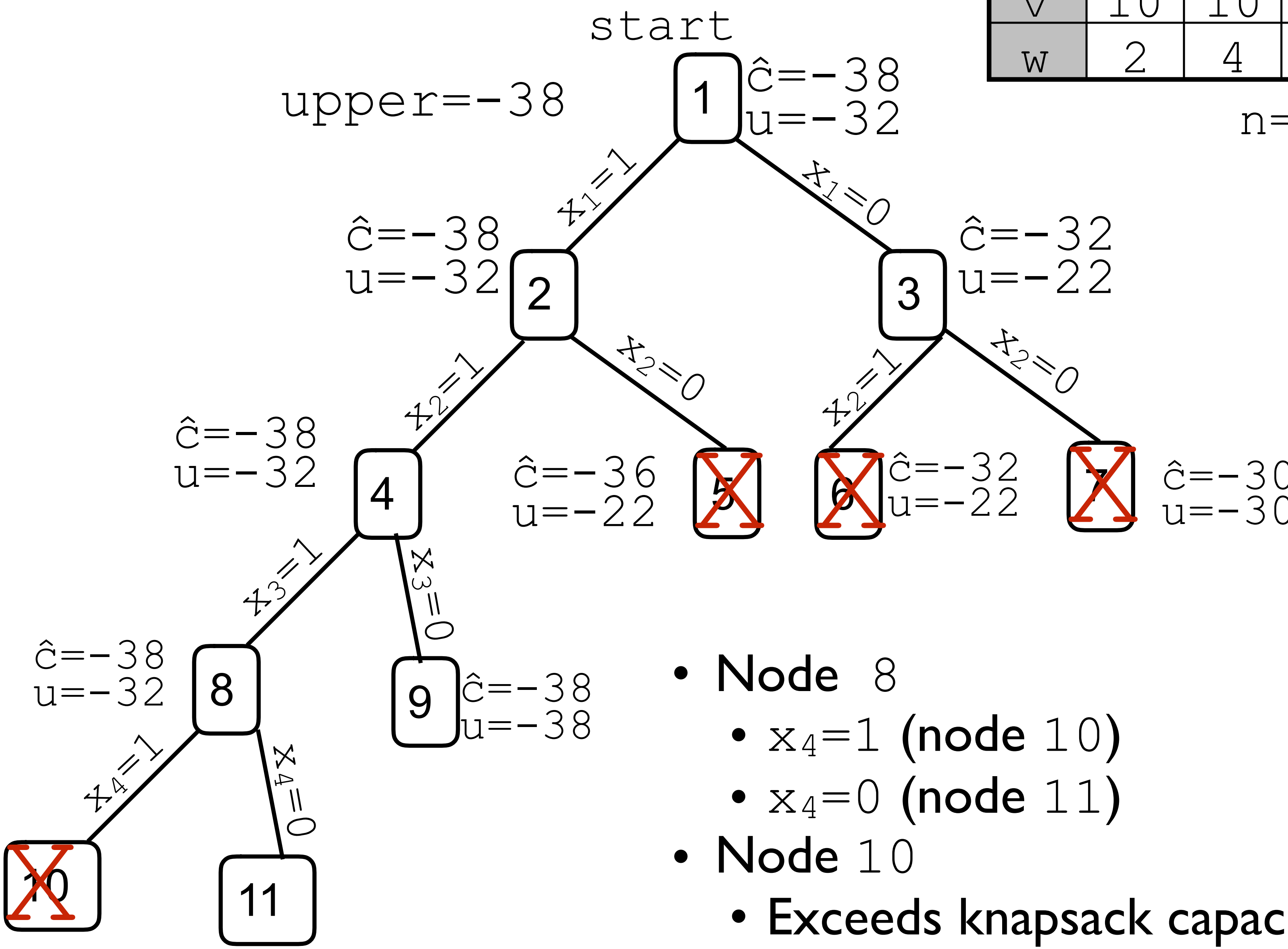


- Since `upper` is updated to `-38`,
  - Nodes `5, 6` are killed, as
  - their cost `ĉ(x)>upper`
- Thus, live nodes are `{8,9}`
- Next E-node to explore: `8`

FIFOBB: 0-1 Knapsack

	1	2	3	4
v	10	10	12	18
w	2	4	6	9

n=4,m=15

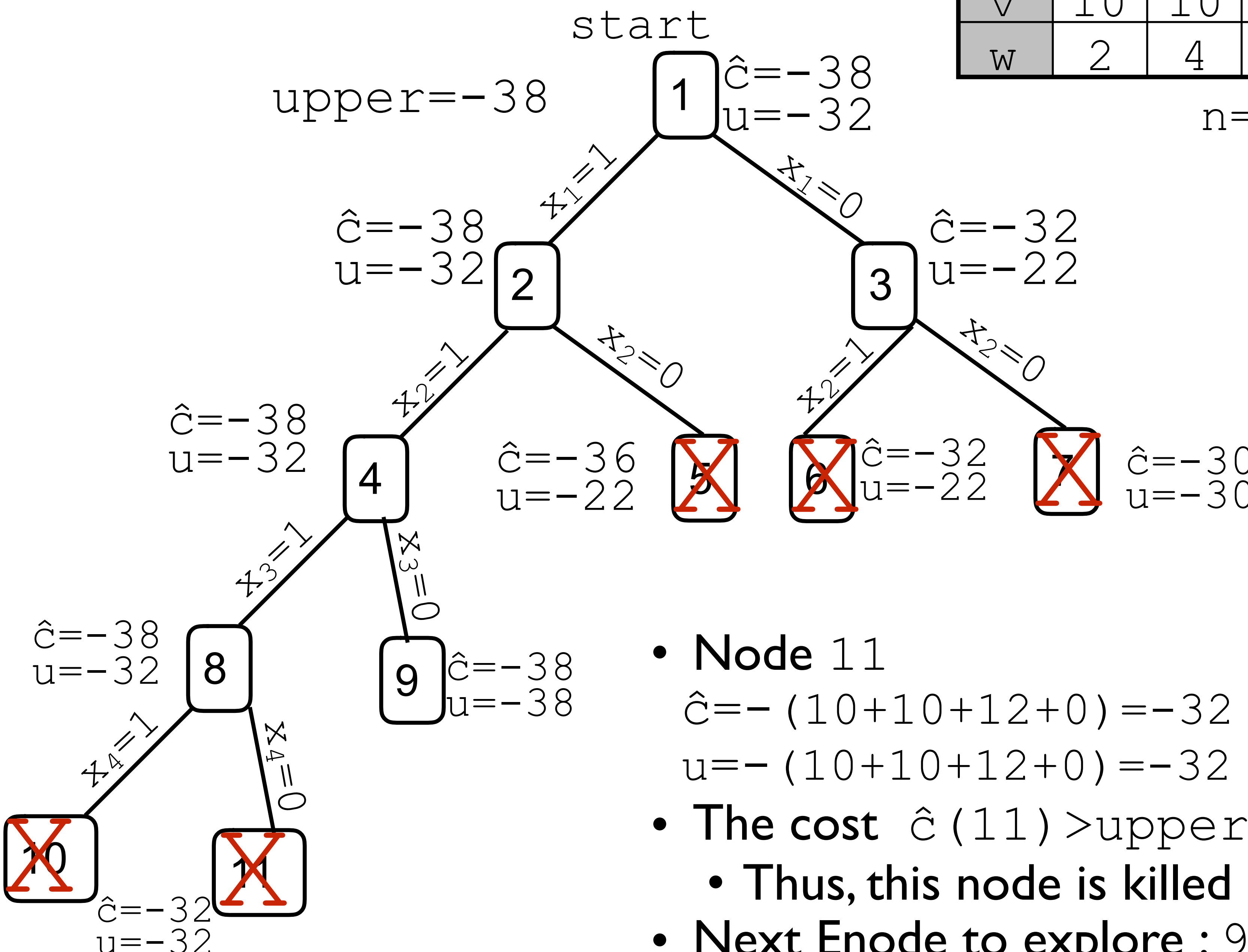


- Node 8
  - x_4=1 (node 10)
  - x_4=0 (node 11)
- Node 10
  - Exceeds knapsack capacity.
  - Can't consider it.

FIFOBB: 0-1 Knapsack

	1	2	3	4
v	10	10	12	18
w	2	4	6	9

n=4, m=15

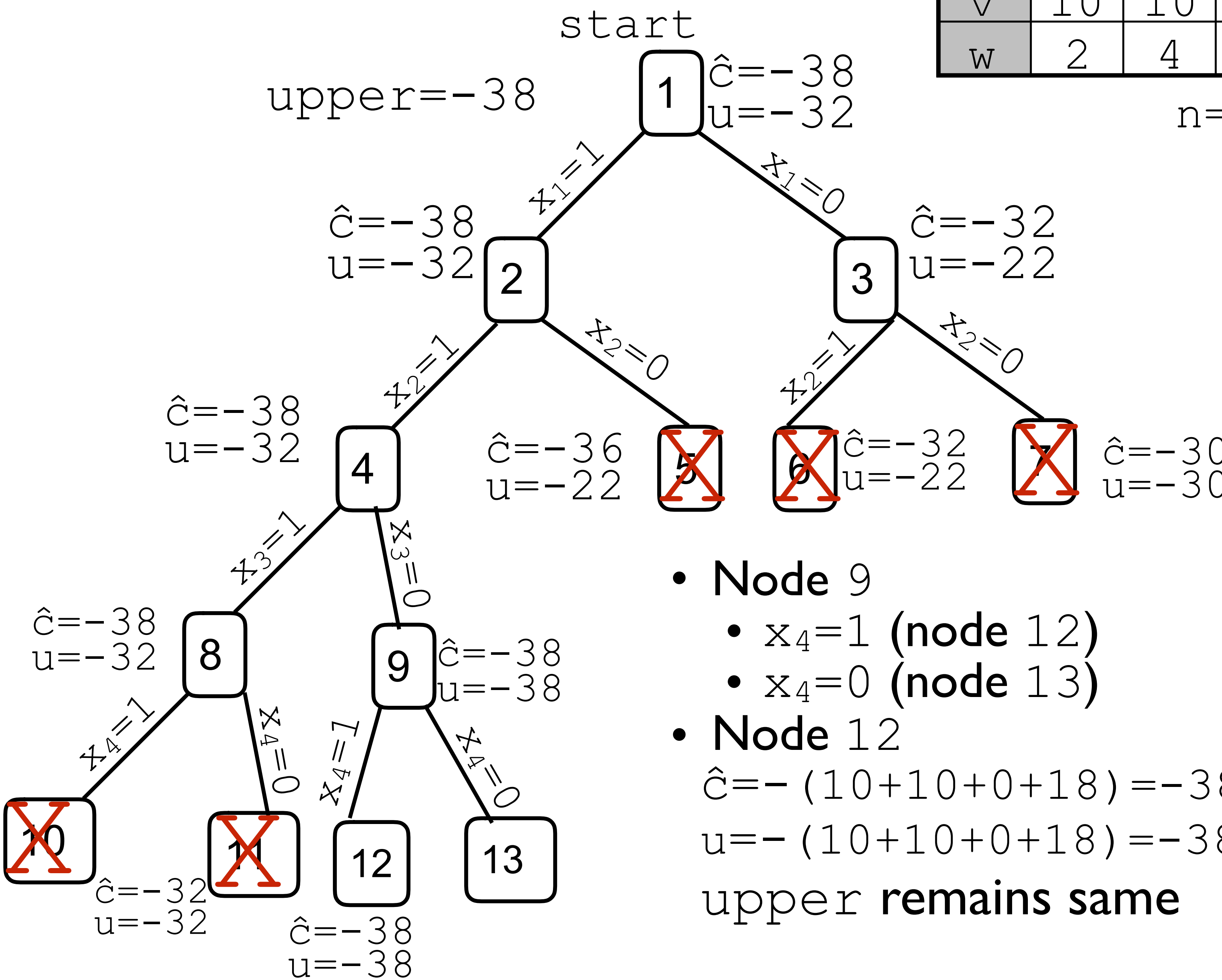


- Node 11
  $\hat{c}=-(10+10+12+0)=-32$
  $u=-(10+10+12+0)=-32$
- The cost $\hat{c}(11) > upper$
  - Thus, this node is killed
- Next Enode to explore : 9

FIFOBB: 0-1 Knapsack

	1	2	3	4
v	10	10	12	18
w	2	4	6	9

n=4,m=15



- **Node** 9
  - $x_4=1$ **(node** 12**)**
  - $x_4=0$ **(node** 13**)**
- **Node** 12

  ĉ=-(10+10+0+18)=-38

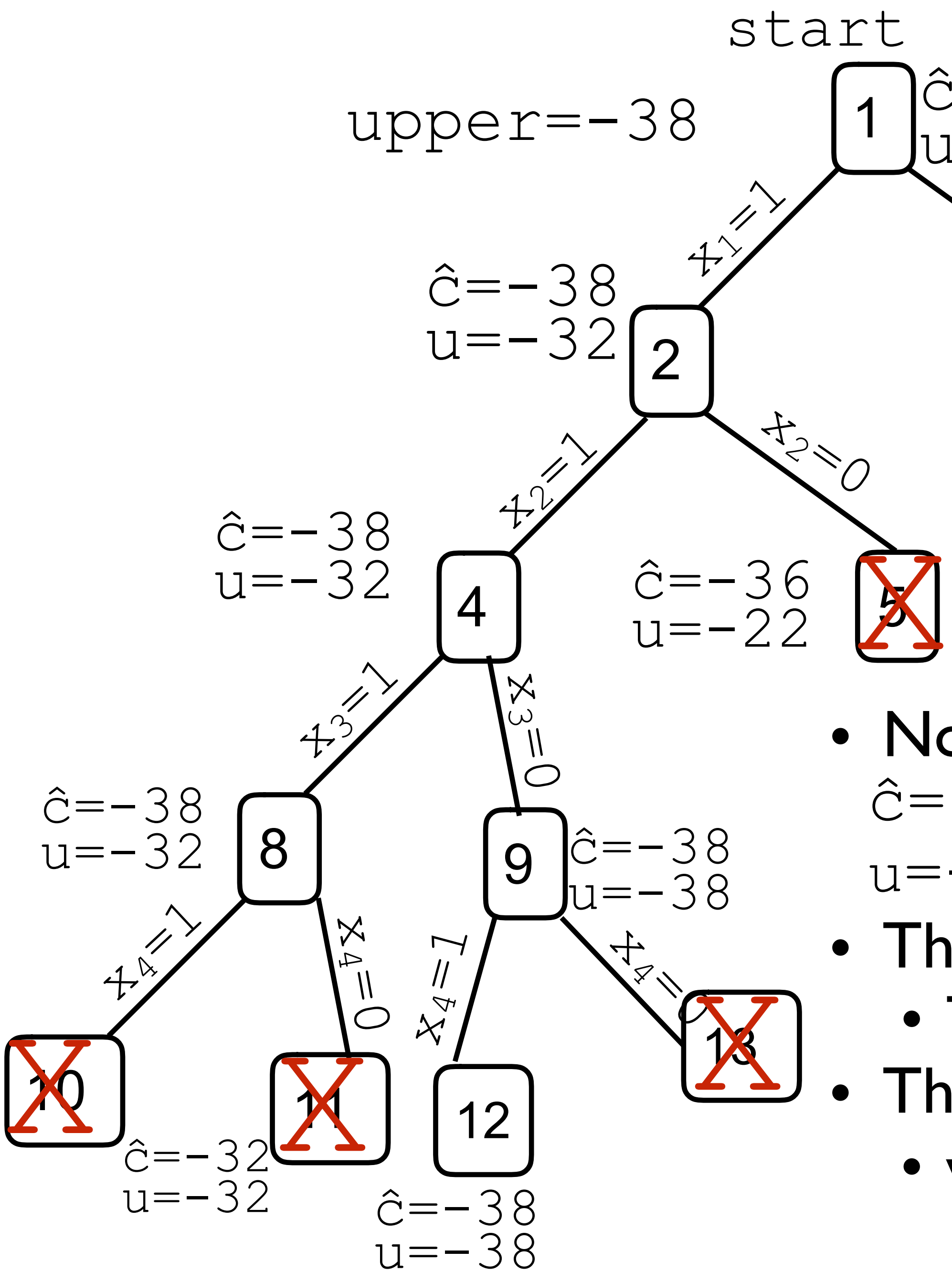
  u=-(10+10+0+18)=-38

  upper **remains same**

FIFOBB: 0-1 Knapsack

	1	2	3	4
v	10	10	12	18
w	2	4	6	9

n=4, m=15

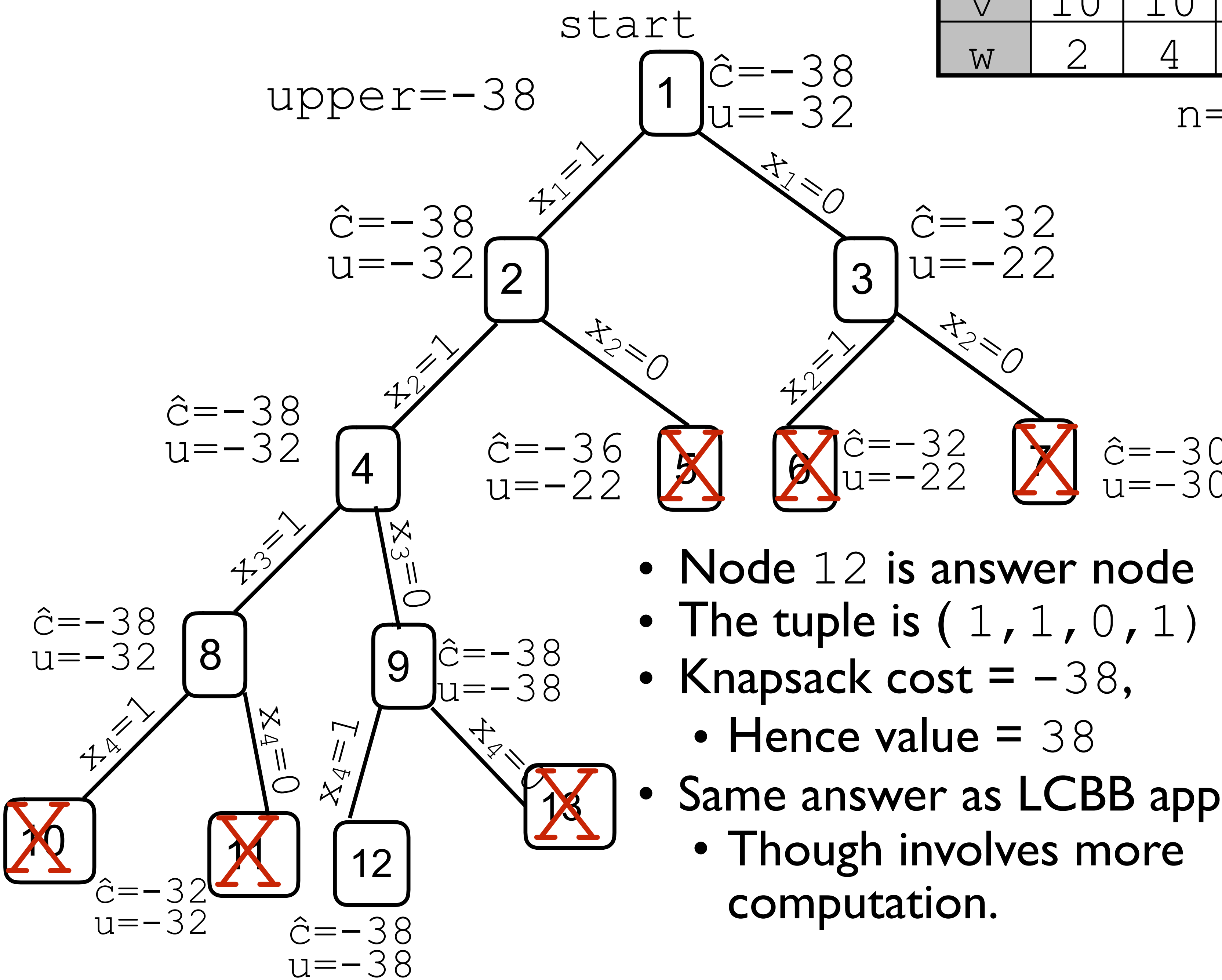


- Node 13
  ĉ=-(10+10+0+0)=-20
  u=-(10+10+0+0)=-20
- The cost ĉ(13)>upper
  - Thus, this node is killed
- The only node left is 12
  - which is answer node

FIFOBB: 0-1 Knapsack

	1	2	3	4
v	10	10	12	18
w	2	4	6	9

n=4,m=15



- Node 12 is answer node
- The tuple is (1,1,0,1)
- Knapsack cost = −38,
  - Hence value = 38
- Same answer as LCBB approach
  - Though involves more computation.

Summary:

- FIFO Branch and Bound for `0-1` Knapsack problem
- Need to do more (exploration) computation as comparted to LCBB.