Adversarial and Problem Reduction Search

Searching in Game Trees and AND/OR Graphs

COURSE: CS60045

Pallab Dasgupta
Professor,
Dept. of Computer Sc & Engg

One of the Defining Moments of AI: Kasparov versus Deep Blue

On May 11, 1997, it won a 6-game match by 2 wins to 1 with 3 draws

Today, we have power to evaluate more than 200 million moves per second !!

Optimization Problems and Games

In the real world multiple agents may compete

- Each agent wants to maximize its payoff
- The sum of the payoffs may be a constant (zero-sum) or may not be a constant

How can I optimize my payoff in the presence of opponents?

- At each state of this game, the profits are shared among the agents (players)
- Each agent wants to take actions (in turns) to reach a state which maximizes its payoff
- A state of the game where none of the agents can increase their profit by taking any of their available actions is a steady state (saddle point)

We assume that each player will look only at its own profit

- There is no collaboration among the players (unfortunately)

A Classical Problem: Prisoner's Dilemma

- Two members of a criminal gang are arrested and imprisoned. Each prisoner is in solitary confinement with no means of communicating with the other.
- The prosecutors lack sufficient evidence to convict the pair on the principal charge, but they have enough to convict both on a lesser charge.
- Simultaneously, the prosecutors offer each prisoner a bargain.

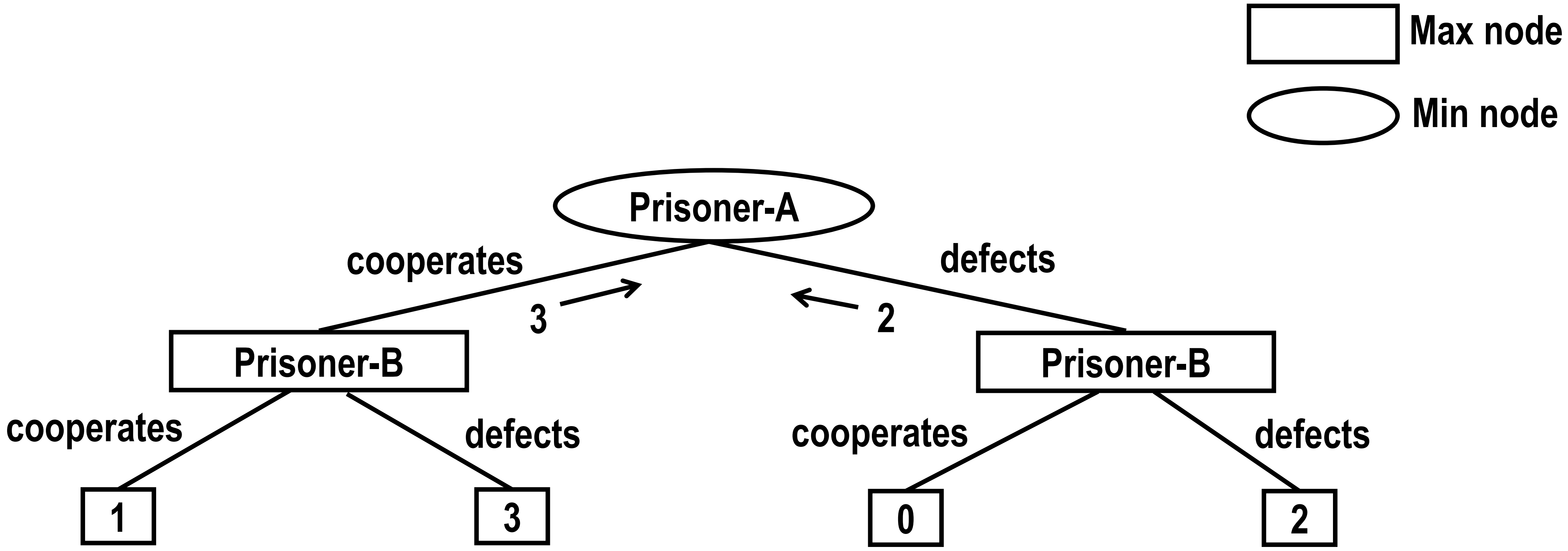
		PRISONER B	
		Prisoner B stays silent (cooperates)	Prisoner B betrays (defects)
PRISONER A	Prisoner A stays silent (cooperates)	Each serves 1 year	Prisoner A: 3 yrs Prisoner B: goes free
	Prisoner A betrays (defects)	Prisoner A: goes free Prisoner B: 3 yrs	Each serves 2 yrs

A Classical Problem: Prisoner’s Dilemma

- Prisoner A will defect. Why?
  - Prisoner-A knows that Prisoner-B may cooperate or defect
    - If Prisoner-B cooperates, then by defecting, Prisoner-A will go free
    - If Prisoner-B defects, then Prisoner-A will face a longer sentence by staying silent
  - In both cases Prisoner-A gains by defecting
  - Therefore both prisoners will defect, although they would have gained from cooperating

		PRISONER B	
		Prisoner B stays silent (cooperates)	Prisoner B betrays (defects)
PRISONER A	Prisoner A stays silent (cooperates)	Each serves 1 year	Prisoner A: 3 yrs Prisoner B: goes free
	Prisoner A betrays (defects)	Prisoner A: goes free Prisoner B: 3 yrs	Each serves 2 yrs

Prisoner's Dilemma



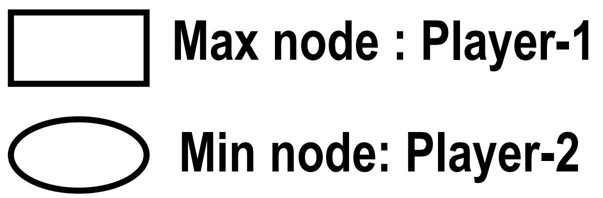
Since Prisoner-A assumes that Prisoner-B is his adversary, he decides to defect, that is, the move to the right is taken at the root

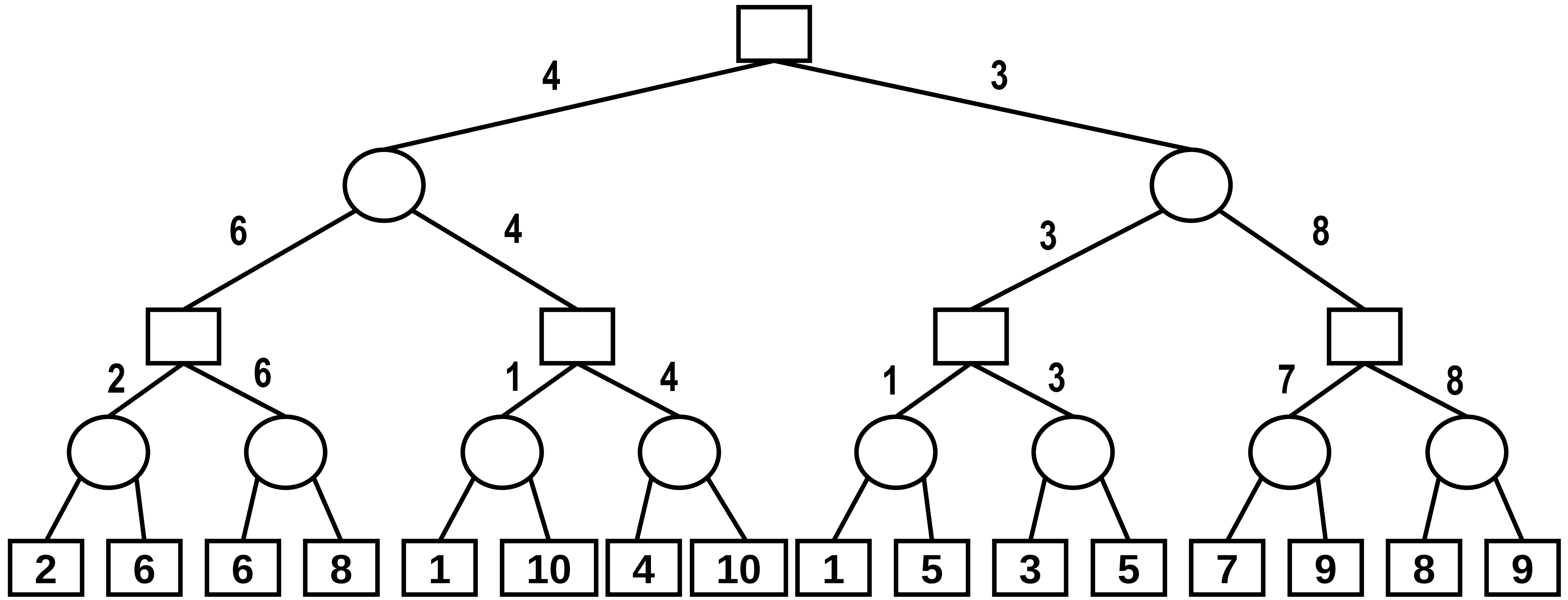
Game Tree

- A tree with three types of nodes, namely Terminal nodes, Min nodes and Max nodes. Terminal nodes have no children. The tree has alternating levels of Max and Min nodes, representing the turns of Player-1 and Player-2 in making moves
- All nodes represent some state of the game
- Terminal nodes are labeled with the payoff for Player-1. It could be Boolean (such as WON or LOST). In large games, where looking ahead up to the WON / LOST states is not feasible, the payoff at a terminal node may represent a heuristic cost representing the quality of the state of the game from Player-1's perspective
- The payoff at a Min node is the minimum among the payoffs of its successors
- The payoff at a Max node is the maximum among the payoffs of its successors

- If Player-1 aims to maximize its payoff, then it represents Max nodes, else it represents Min nodes.

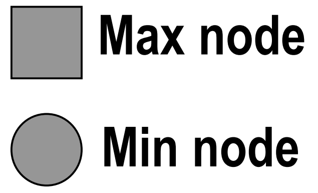
Finding the payoff at the root node

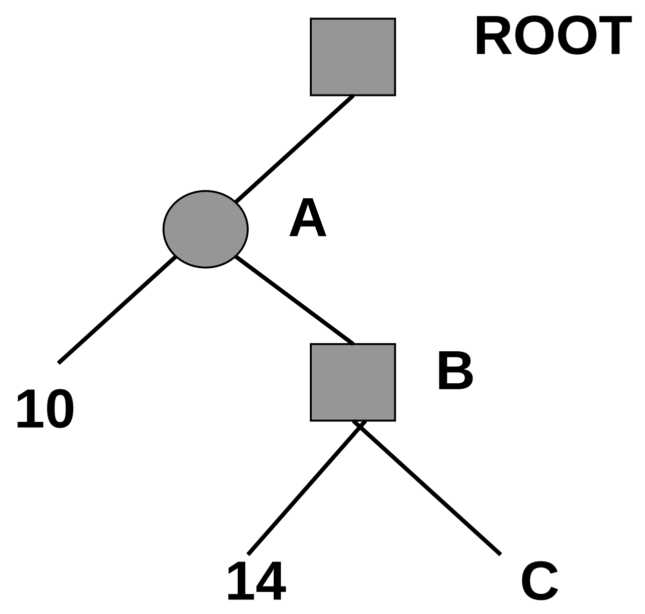




Therefore, the left move is taken at the root

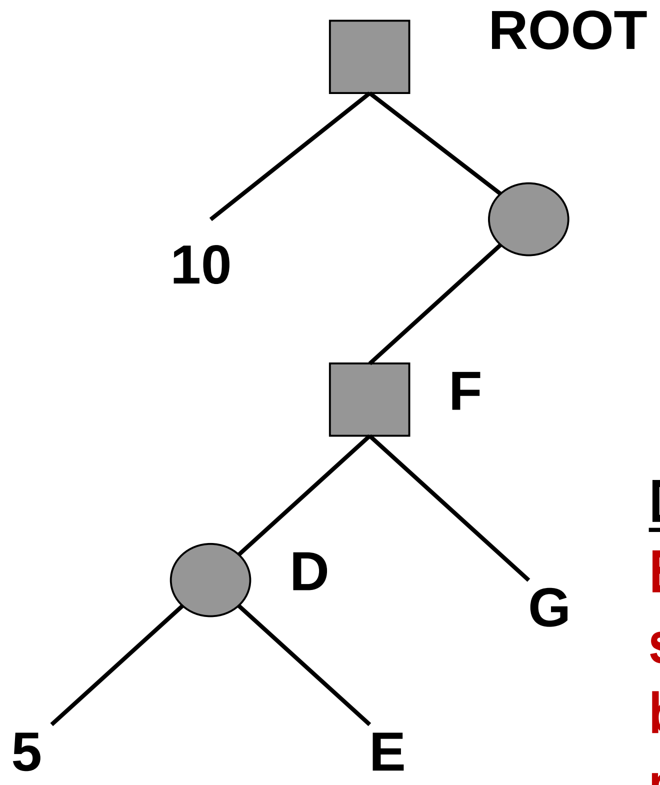
Shallow and Deep Pruning





Shallow Cut-off

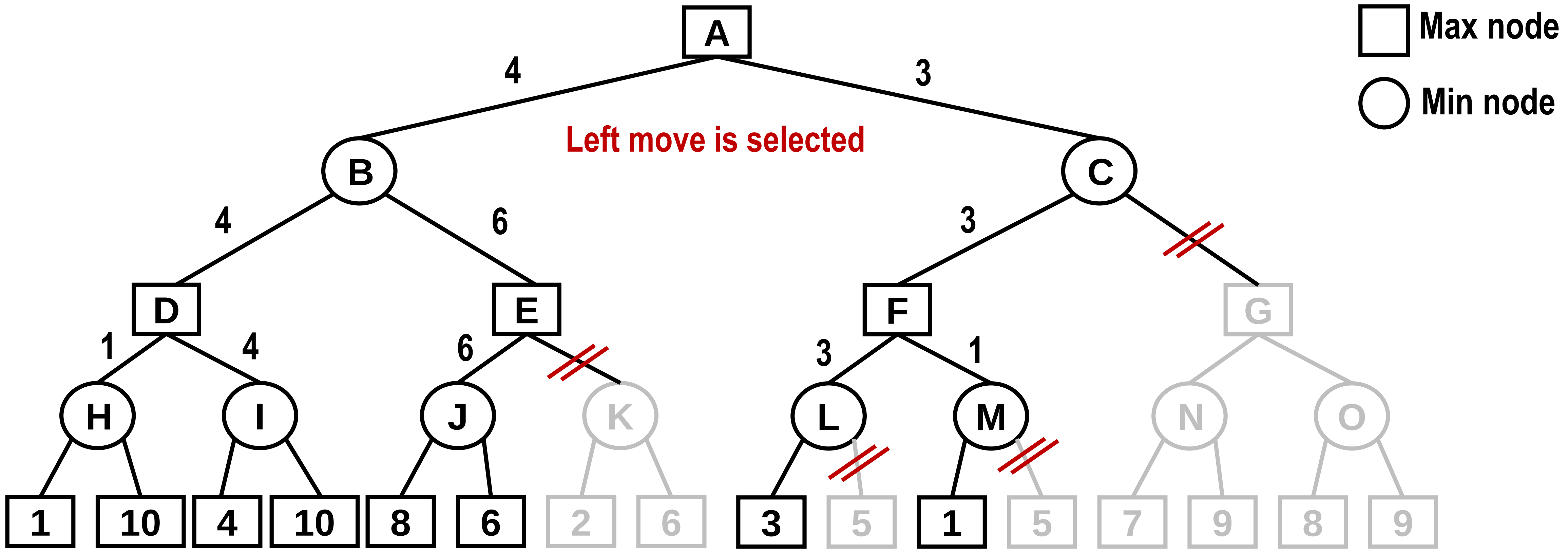
Exploring node C and its successors is meaningless because payoff at B is at least 14, and hence A will never choose the move to B.



Deep Cut-off

Exploring node E and its successors is meaningless because the payoff at node D is at most 5, whereas the ROOT can already guarantee a payoff of 10 by choosing the left move. Therefore, the game will never reach the node D.

Pruning explained



- Node K is pruned because E will have cost at least 6 and B already has a child of cost 4
- Right child of node M is pruned because M will have cost of at most 1 and F already has a child of cost 3
- Node G is pruned because C will have cost of at most 3 and A already has a child of cost 4

Alpha-Beta Pruning

Alpha Bound of J:

- The max current payoff of all MAX ancestors of J
- Exploration of a min node, J, is stopped when its payoff equals or falls below alpha
- In a min node, we update beta

Beta Bound of J:

- The min current payoff of all MIN ancestors of J
- Exploration of a max node, J, is stopped when its payoff equals or exceeds beta
- In a max node, we update alpha

In both min and max nodes, we return when $\alpha \geq \beta$

Alpha-Beta Pruning Procedure: $V(J;\alpha,\beta)$

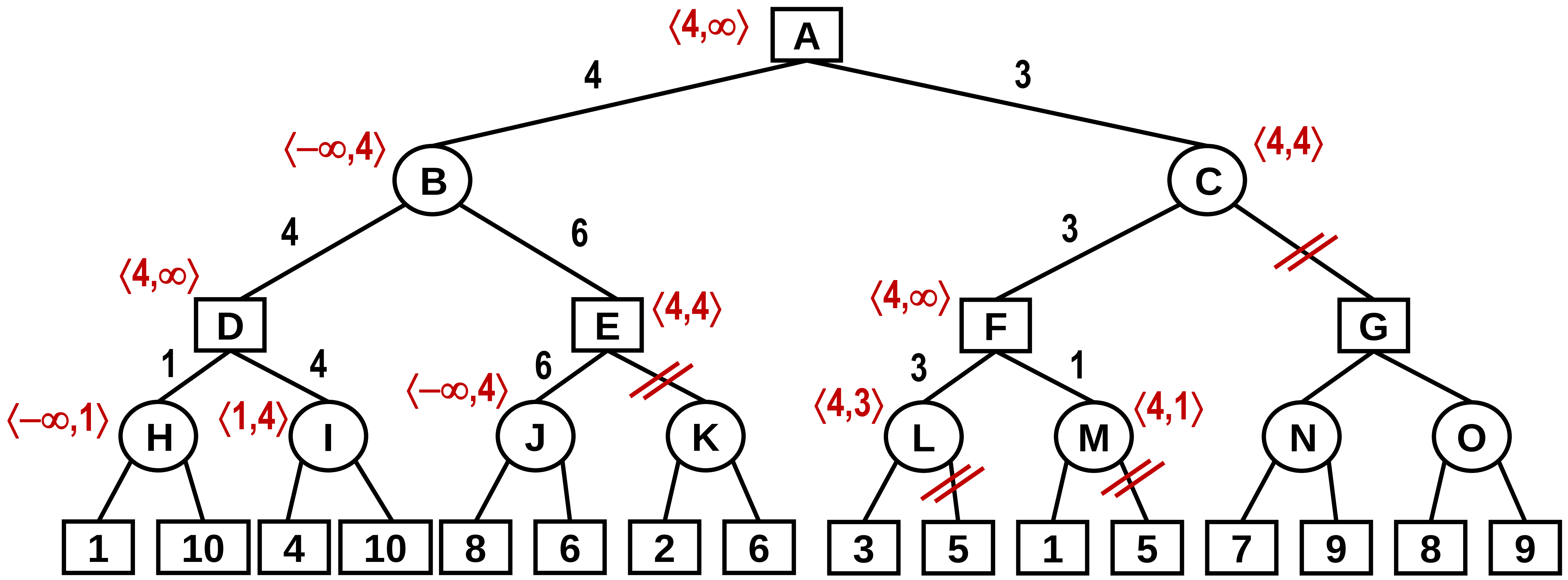
The initial call is with $V(\text{Root}; -\infty, +\infty)$

```
1.  If J is a terminal, return V(J) = h(J).

2.  If J is a max node:
        For each successor $J_k$ of J in succession:
            Set $\alpha = \max\{\alpha, V(J_k; \alpha, \beta)\}$
            If $\alpha \geq \beta$ then return $\beta$, else continue
        Return $\alpha$

3.  If J is a min node:
        For each successor $J_k$ of J in succession:
            Set $\beta = \min\{\beta, V(J_k; \alpha, \beta)\}$
            If $\alpha \geq \beta$ then return $\alpha$, else continue
        Return $\beta$
```

Pruning showing ⟨α, β⟩ values



- The vectors shown besides the nodes are the ⟨ α, β ⟩ values at the time of return from that node
- Node K is pruned because α = β
- Right child of node M is pruned because α > β
- Node G is pruned because α = β

Games in AI

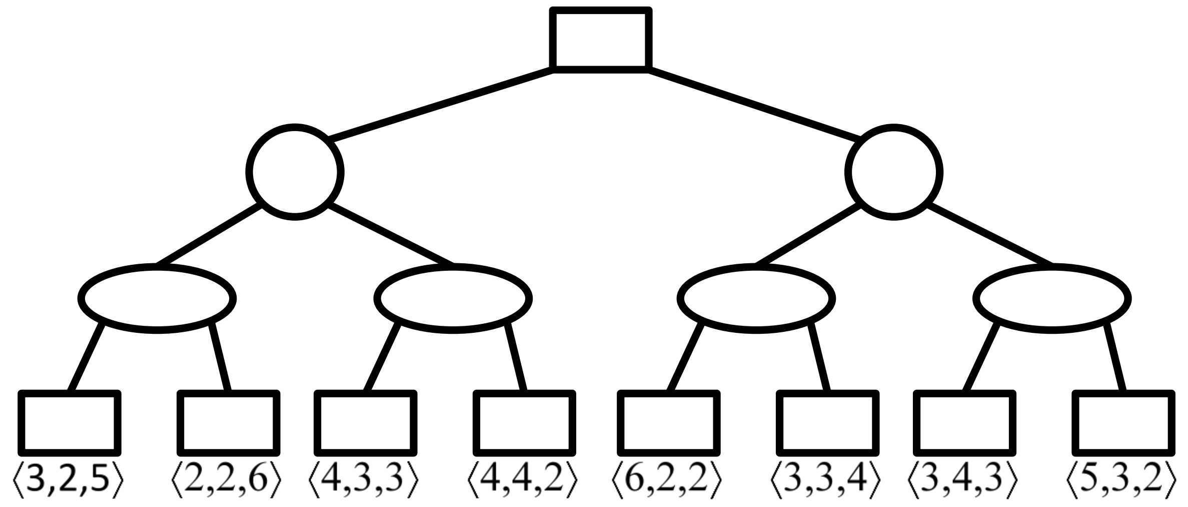
- The use of AI in playing computer games is well known

- Games are also ubiquitous in the real world. Areas include:
  - Economics – the original bastion of game theory
  - Reactive systems such as control systems acting in safety critical environments
  - Autonomous driving

Exercise Problem

The following game tree is for a three player game. The vector ⟨*x, y, z*⟩ in each terminal node represents the payoffs for Player-1, Player-2, and Player-3 respectively. The square nodes represent moves for Player-1, the circular nodes represent moves of Player-2 and the oval nodes represent moves for Player-3.

The three player game may assume that the opponents are rational, that is, each player's sole motive is to maximize the payoff of that player. Under this assumption, show the preferred choice of successor at each node by marking the relevant edges.

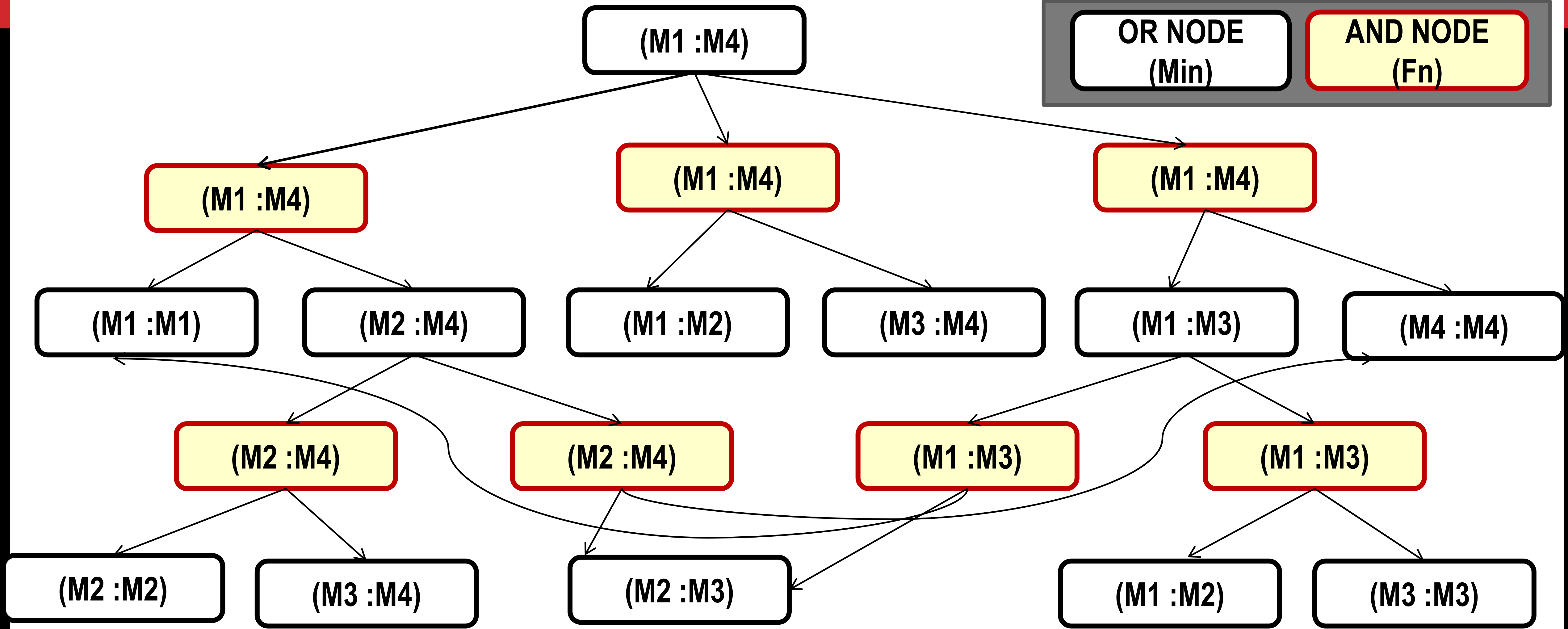


A more conservative approach for Player-1 is to choose the move without making the assumption of rational opponents. This situation allows Player-2 and Player-3 to conspire to minimize the payoff of Player-1. If this is the case, then on the same game tree show the preferred choice of successor at each node by marking the relevant edges.

Problem Reduction Search

- Planning how best to solve a problem that can be recursively decomposed into sub-problems in multiple ways
    - Matrix multiplication problem
    - Tower of Hanoi
    - Blocks World problems
    - Theorem proving

COMPOSITIONAL AND/OR GRAPHS: MATRIX CHAIN MULTIPLICATION



(M1 X (M2 X (M3 X M4))) = ((M1 x M2) X (M3 X M4)) = (((M1 x M2) X M3) X M4) = (M1 X (M2 X M3)) X M4)

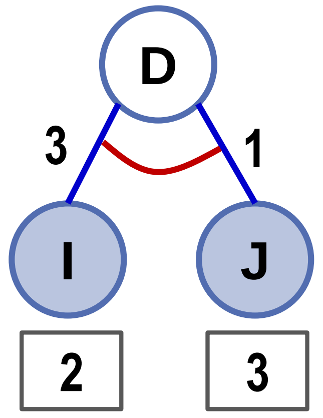
BUT THE NUMBER OF MULTIPLICATIONS TO GET THE ANSWER DIFFER !!

AND / OR Trees

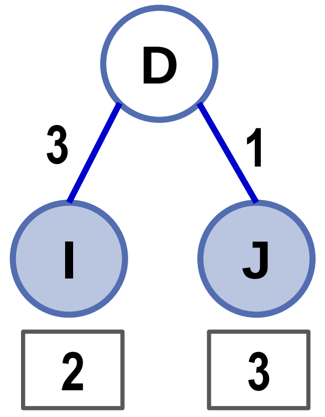
- Two types of nodes:
  - AND Nodes – Every children will have to be solved
    - Cost of solving problem D is:

      3 + Cost of solving I + 1 +Cost of solving J

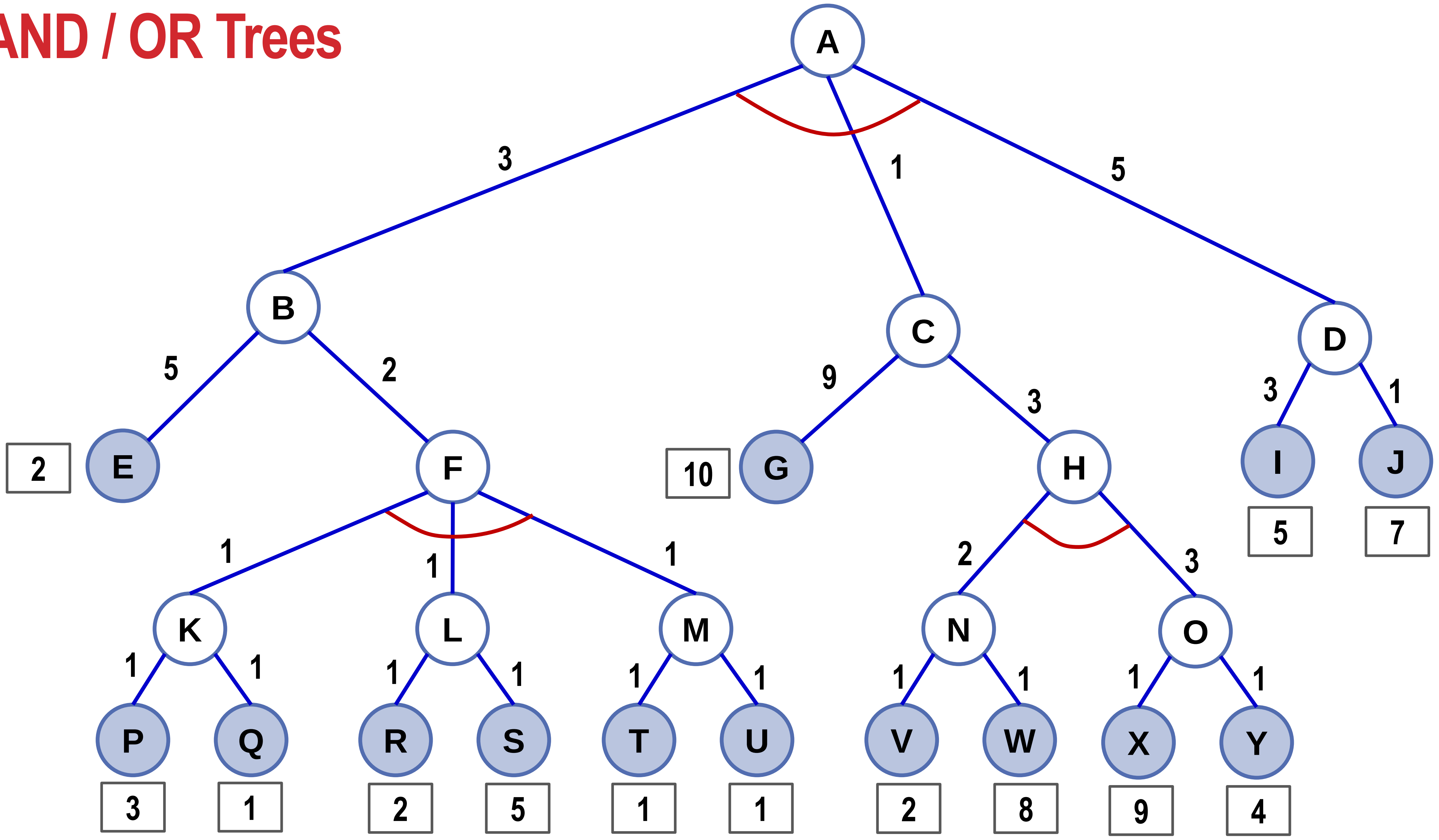
      = 3+2 + 1+3 = 9



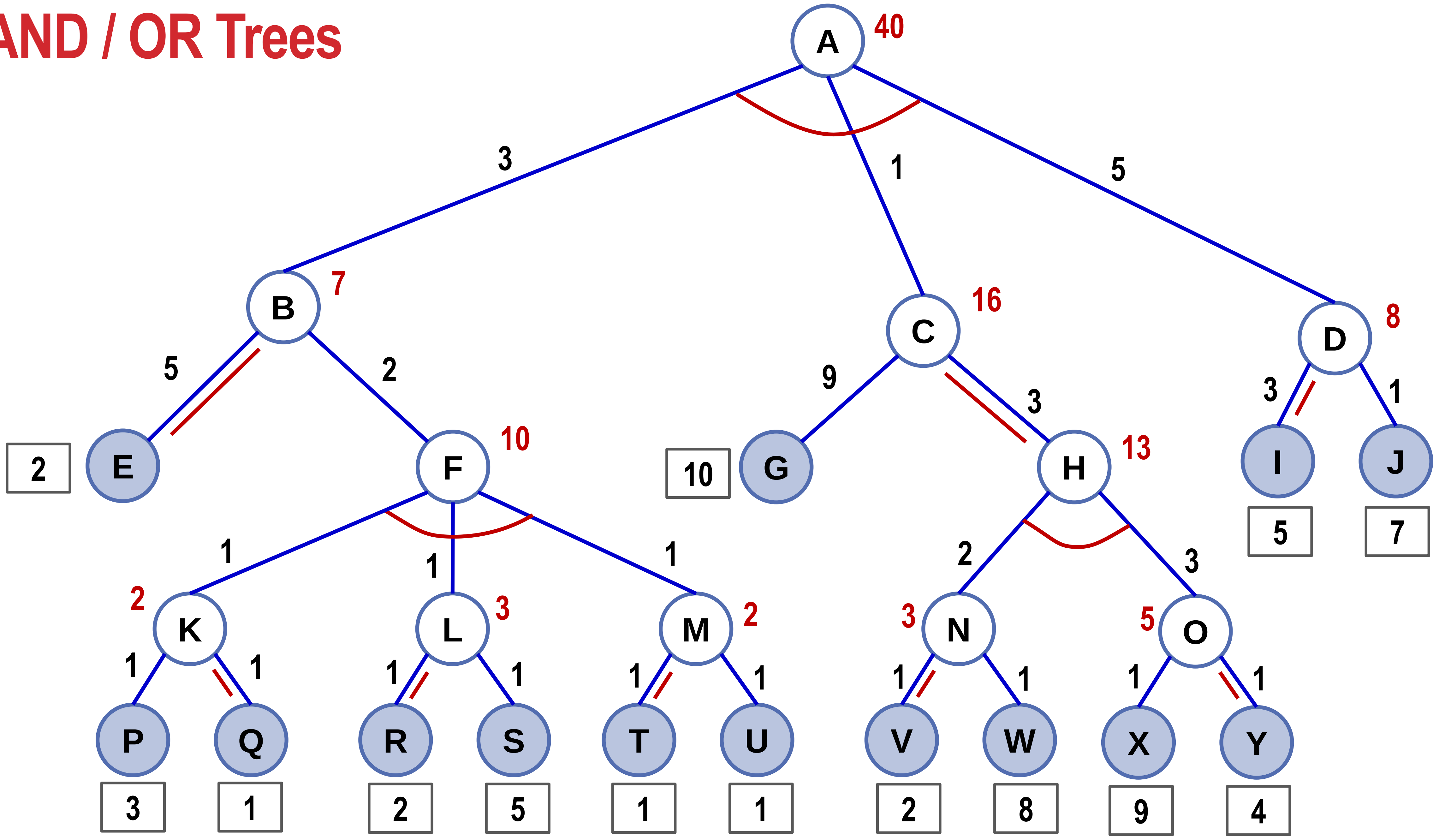
  - OR Nodes – One of the children will have to be solved
    - Cost of solving D by reducing it to I is 3 + 2 = 5
    - Cost of solving D by reducing it to J is 1 + 3 = 4
    - To solve problem D it is better to reduce it to J



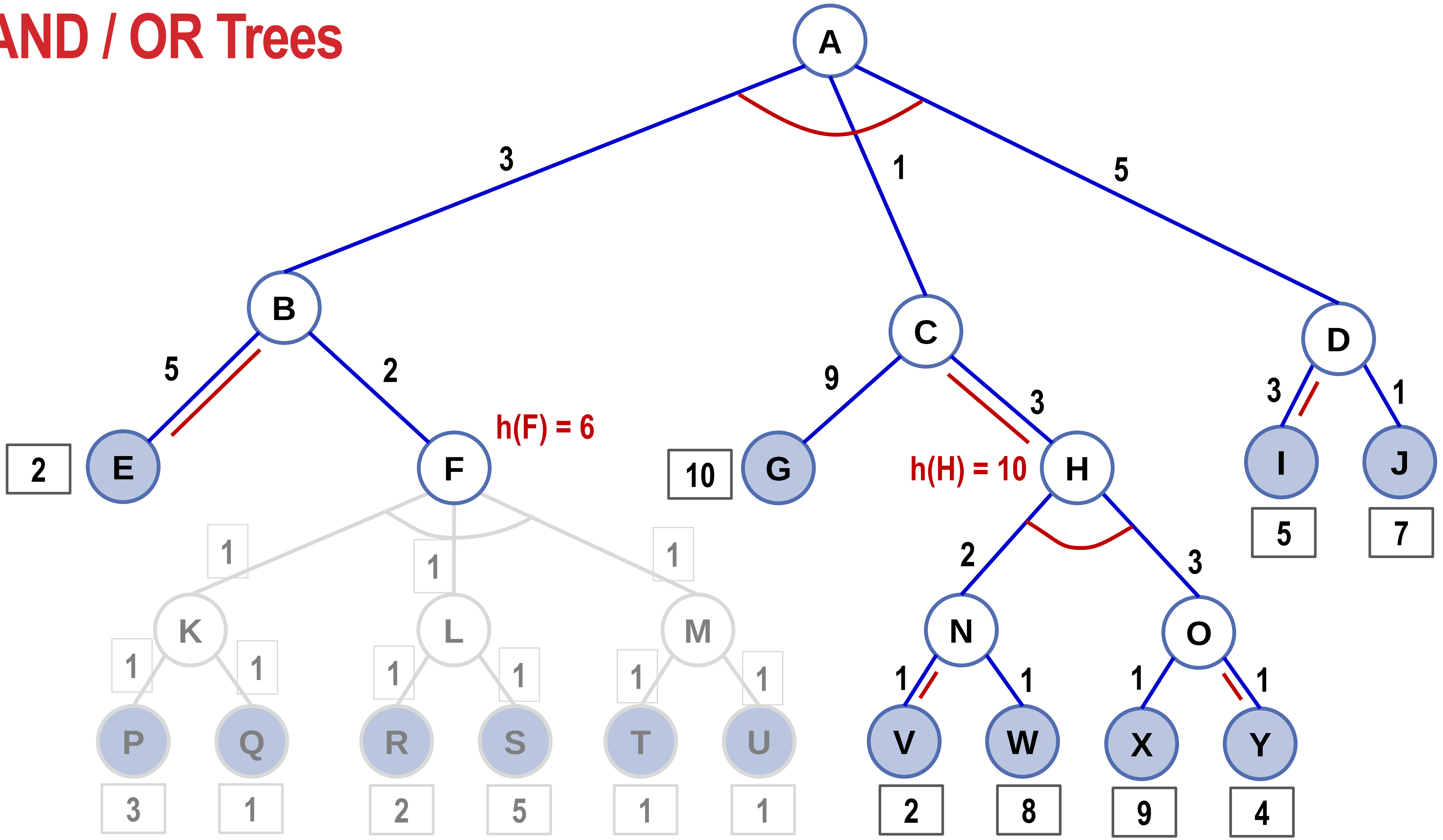
AND / OR Trees


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
3
1
5
5
2
9
3
3
1
1
1
1
2
3
1
1
1
1
1
1
1
1
1
1
1
1
2
10
5
7
3
1
2
5
1
1
2
8
9
4

AND / OR Trees


A
40
3
1
5
B
7
C
16
D
8
5
2
9
3
3
1
2
E
F
10
10
G
H
13
I
J
5
7
1
1
1
2
3
2
K
L
3
M
2
3
N
5
O
1
1
1
1
1
1
1
1
1
1
P
Q
R
S
T
U
V
W
X
Y
3
1
2
5
1
1
2
8
9
4

AND / OR Trees


A
B
C
D
3
1
5
5
2
9
3
3
1
2
E
F
10
G
h(F) = 6
h(H) = 10
H
I
J
5
7
1
1
1
1
K
L
M
1
1
1
1
1
1
P
Q
R
S
T
U
3
1
2
5
1
1
2
3
N
O
1
1
1
1
V
W
X
Y
2
8
9
4

The AND/OR graph search problem

- **Problem definition:**
  - Given: [G, s, T] where
    - G: implicitly specified AND/OR graph
    - S: start node of the AND/OR graph
    - T: set of terminal nodes
    - h(n) heuristic function estimating the cost of solving the sub-problem at n
  - To find:
    - A minimum cost solution tree

Algorithm AO*

```
1. Initialize:        Set $G^* = \{s\}$, $f(s) = h(s)$
                          If $s \in T$, label s as SOLVED
2. Terminate:         If s is SOLVED, then Terminate
3. Select:            Select a non-terminal leaf node n from the marked sub-tree
4. Expand:            Make explicit the successors of n
                          For each new successor, m:
                              Set $f(m) = h(m)$
                          If m is terminal, label m SOLVED
5. Cost Revision:     Call cost-revise(n)
6. Loop:              Go To Step 2.
```

Cost Revision in AO*: cost-revise(n)

```
1. Create Z = {n}
2. If Z = { } return
3. Select a node m from Z such that m has no
   descendants in Z
4. If m is an AND node with successors
   $r_1, r_2, \ldots r_k$:
      Set $f(m) = \Sigma\ [\ f(r_i) + c(m, r_i)\ ]$
      Mark the edge to each successor of m
      If each successor is labeled SOLVED,
          then label m as SOLVED
5. If m is an OR node with successors
   $r_1, r_2, \ldots r_k$:
      Set $f(m) = \min \{ f(r_i) + c(m, r_i) \}$
      Mark the edge to the best successor of m
      If the marked successor is labeled
      SOLVED, label m as SOLVED
6. If the cost or label of m has changed,
   then insert those parents of m into Z for
   which m is a marked successor
7. Go to Step 2.
```