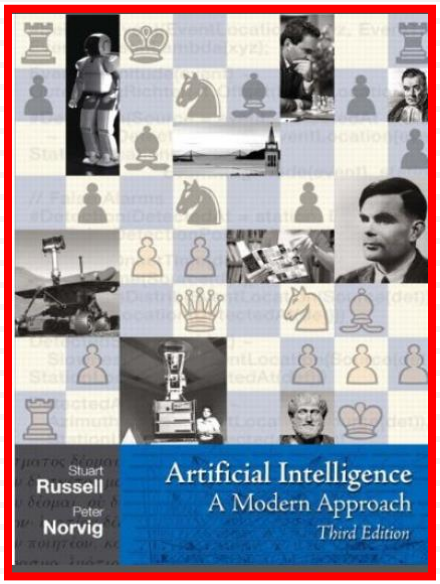
Princess Nora University
Faculty of Computer & Information Systems

جامعة الأميرة نورة بنت عبد الرحمن
Princess Nora Bint Abdul Rahman University

ARTIFICIAL INTELLIGENCE

(CS 370D)

جامعة الأميرة نورة بنت عبد الرحمن
Princess Nora Bint Abdul Rahman University



College of Computer and Information Sciences

(CHAPTER-5)
ADVERSARIAL SEARCH

ADVERSARIAL SEARCH

- Optimal decisions
- MinMax algorithm
- α-β pruning
- Imperfect, real-time decisions

Dr. Abeer Mahmoud
(course coordinator)

ADVERSARIAL SEARCH

Dr. Abeer Mahmoud
(course coordinator)

- Search problems seen so far:
  - ✓ Single agent.
  - ✓ No interference from other agents and no competition.

- Game playing:
  - ✓ Multi-agent environment.
  - ✓ Cooperative games.
  - ✓ Competitive games → **adversarial search**

- **Specifics of adversarial search:**
  - ✓ Sequences of player's decisions we control.
  - ✓ Decisions of other players we do not control.

Dr. Abeer Mahmoud
(course coordinator)

Games vs. search problems

- "Unpredictable" opponent → specifying a move for every possible opponent reply
- Time limits → unlikely to find goal, must approximate

Game as Search Problem
MinMax Algorithm

Dr. Abeer Mahmoud
(course coordinator)

Game setup

- **Consider a game with Two players: (Max)** and **(Min)**
- **Max** moves first and they take turns until the game is over. Winner gets award, loser gets penalty.
- **Games as search:**
  - **Initial state**: e.g. board configuration of chess
  - **Successor function**: list of (move, state) pairs specifying legal moves.
  - **Goal test**: Is the game finished?
  - **Utility function**: Gives numerical value of terminal states. E.g. win (+1), lose (-1) and draw (0) in tic-tac-toe

Max uses **search tree** to determine next move.

Dr. Abeer Mahmoud
(course coordinator)

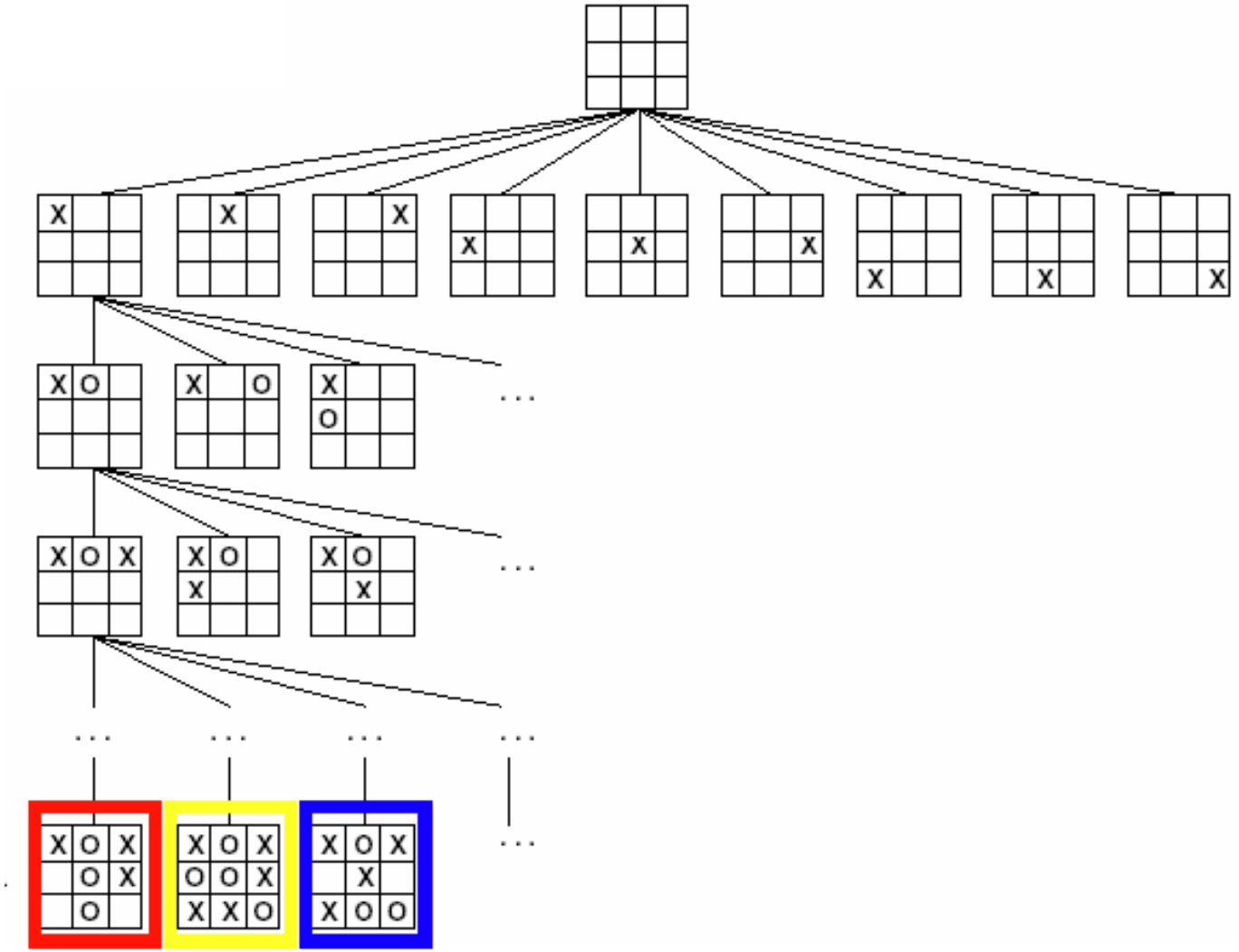
Example of an ADVERSARIAL two player Game

Tic-Tac-Toe (TTT)

- MAX has 9 possible first moves, etc.
- Utility value is always from the point of view of MAX.
- High values good for MAX and bad for MIN.

Dr. Abeer Mahmoud
(course coordinator)

Example of an adversarial 2 person game: Tic-tac-toe



Mahmoud
ordinator)

How to Play a Game by Searching

- **General Scheme**
  - Consider all legal moves, each of which will lead to some new state of the environment ('board position')
  - **Evaluate** each possible resulting board position
  - Pick the move which leads to the best board position.
  - Wait for your opponent's move, then **repeat**.
- **Key problems**
  - Representing the 'board'
  - Representing legal next boards
  - Evaluating positions
  - Looking ahead

Dr. Abeer Mahmoud
(course coordinator)

Game Trees

- Represent the problem space for a game by a tree
  - **Nodes** represent 'board positions' (state)
  - **edges** represent legal moves.

- **Root node** is the position in which a decision must be made.

- **Evaluation function *f*** assigns real-number scores to `board positions.'

- **Terminal nodes (leaf)** represent ways the game could end, labeled with the desirability of that ending (e.g. win/lose/draw or a numerical score)

Dr. Abeer Mahmoud
(course coordinator)

MAX & MIN Nodes

- When I move, I attempt to **MAXimize** my performance.
- When my opponent moves, he attempts to **MINimize** my performance.

TO REPRESENT THIS:

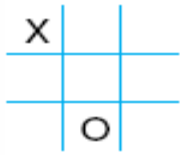
- If we move first, label the root MAX; if our opponent does, label it MIN.
- Alternate labels for each successive tree level.
  - if the root (level 0) is our turn (MAX), all even levels will represent turns for us (MAX), and all odd ones turns for our opponent (MIN).

Evaluation functions

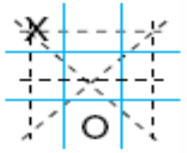
- Evaluations how good a 'board position' is
  - Based on static features of that board alone
- Zero-sum assumption lets us use one function to describe goodness for both players.
  - **f(n)>0** if we are winning in position n
  - **f(n)=0** if position n is tied
  - **f(n)<0** if our opponent is winning in position n
- Build using expert knowledge (Heuristic),
  - Tic-tac-toe: f(n)=(# of 3 lengths possible for me) - (# 3 lengths possible for you)

Dr. Abeer Mahmoud
(course coordinator)

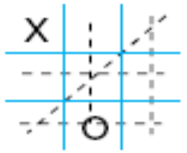
Heuristic measuring for adversarial tic-tac-toe



X has 6 possible win paths:



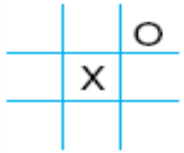
O has 5 possible wins:



E(n) = 6 – 5 = 1

X has 4 possible win paths;
O has 6 possible wins

E(n) = 4 – 6 = –2



X has 5 possible win paths;
O has 4 possible wins

E(n) = 5 – 4 = 1

Heuristic is E(n) = M(n) – O(n)
where M(n) is the total of My possible winning lines
O(n) is total of Opponent's possible winning lines
E(n) is the total Evaluation for state n

Maximize E(n)

E(n) = 0 when my opponent and I have equal number of possibilities.

MinMax Algorithm

- Main idea: choose move to position with highest minimax value. = best achievable payoff against best play.
- E.g., 2-ply game:

Dr. Abeer Mahmoud
(course coordinator)

MinMax Algorithm

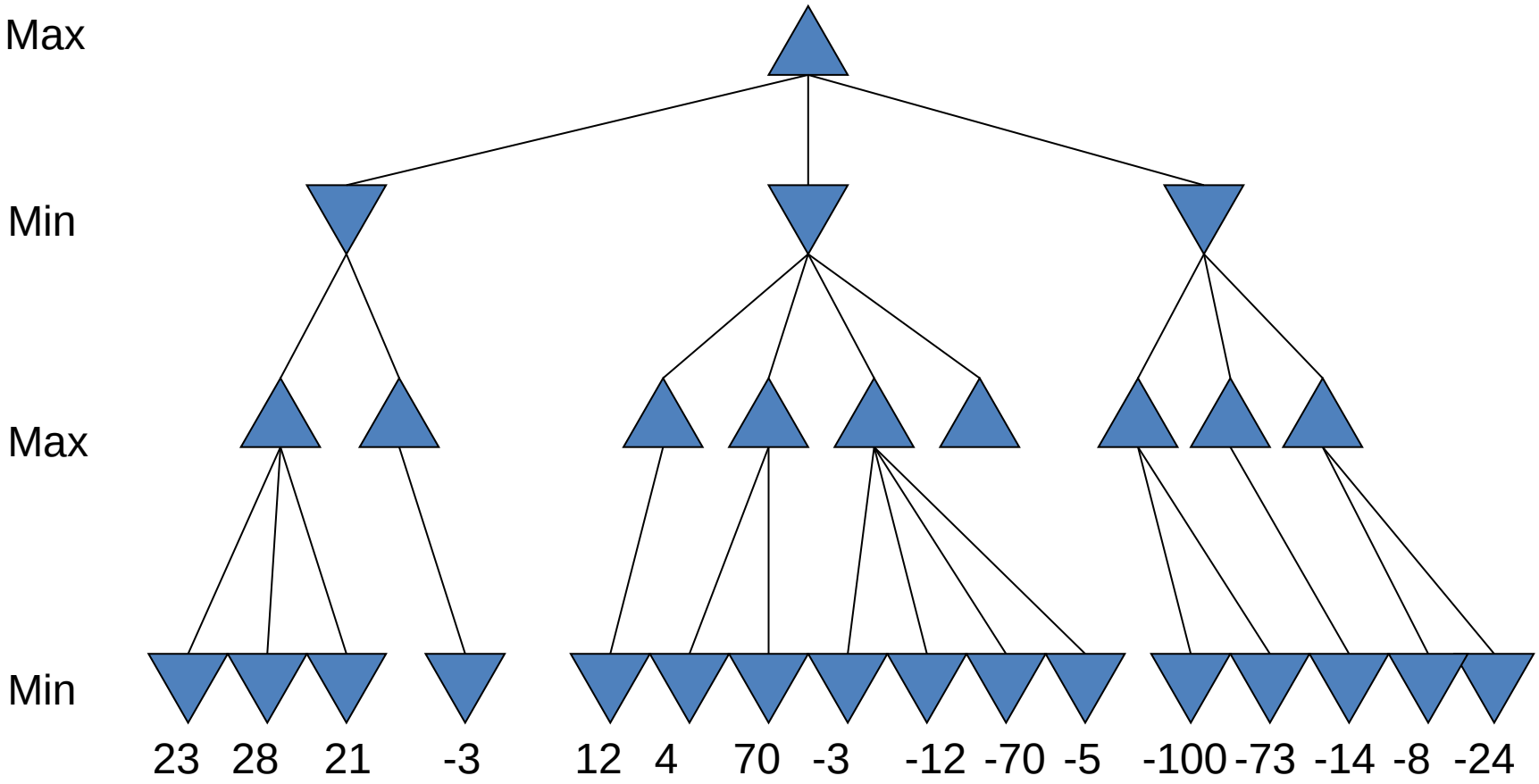
```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s))
    return v
```

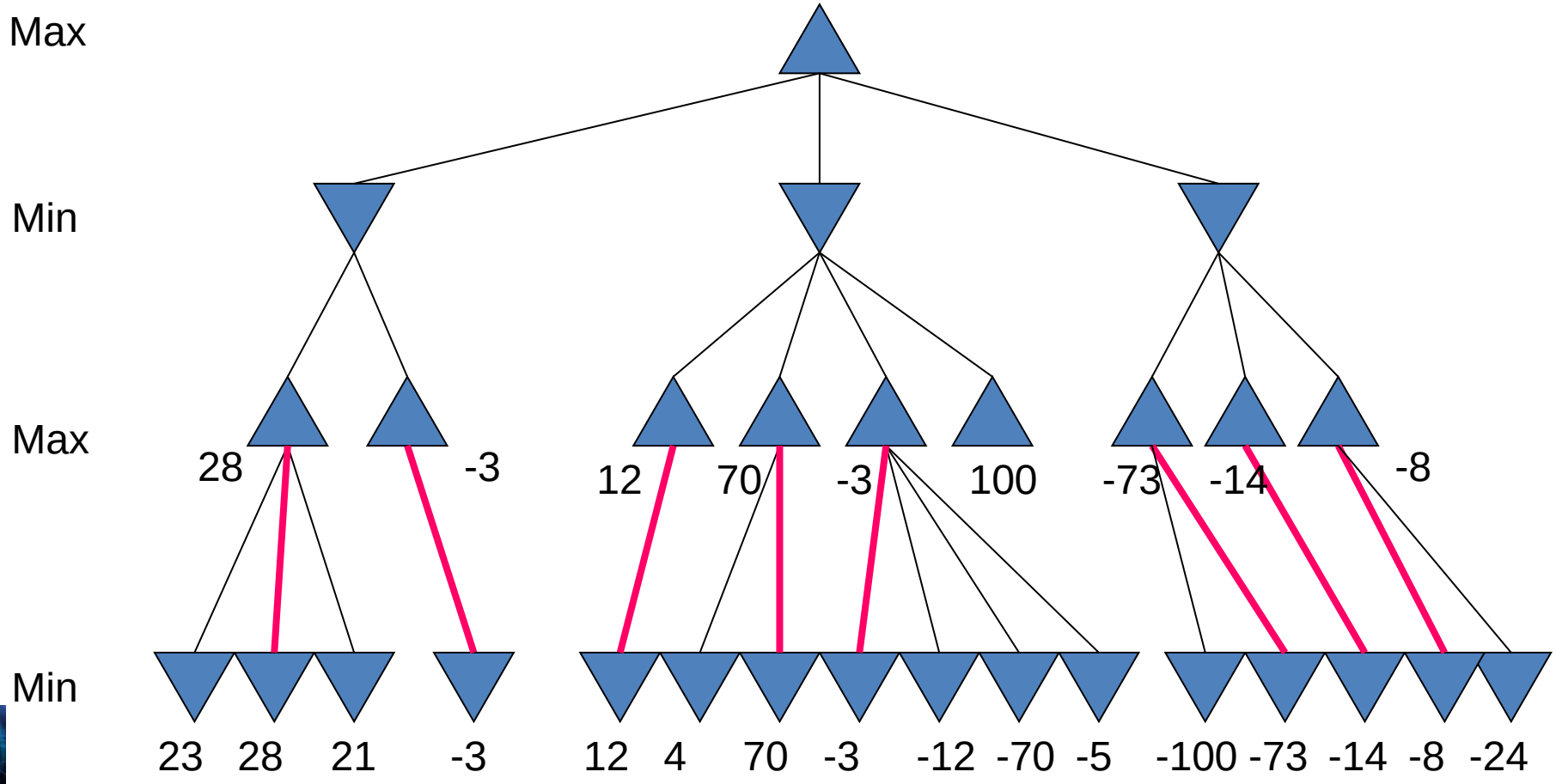
Dr. Abeer Mahmoud
(course coordinator)

Minimax tree



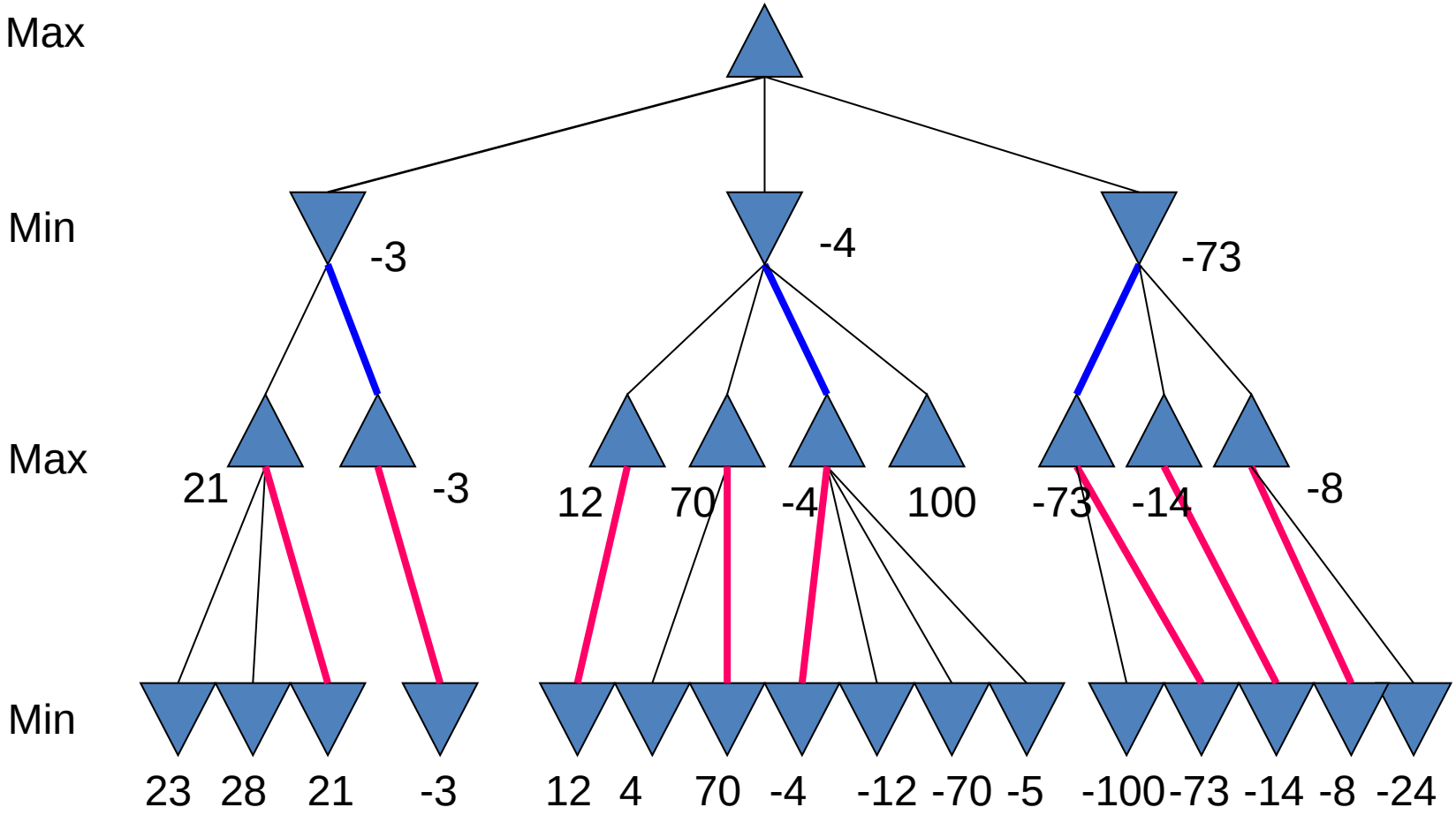
Dr. Abeer Mahmoud
(course coordinator)

Minimax tree



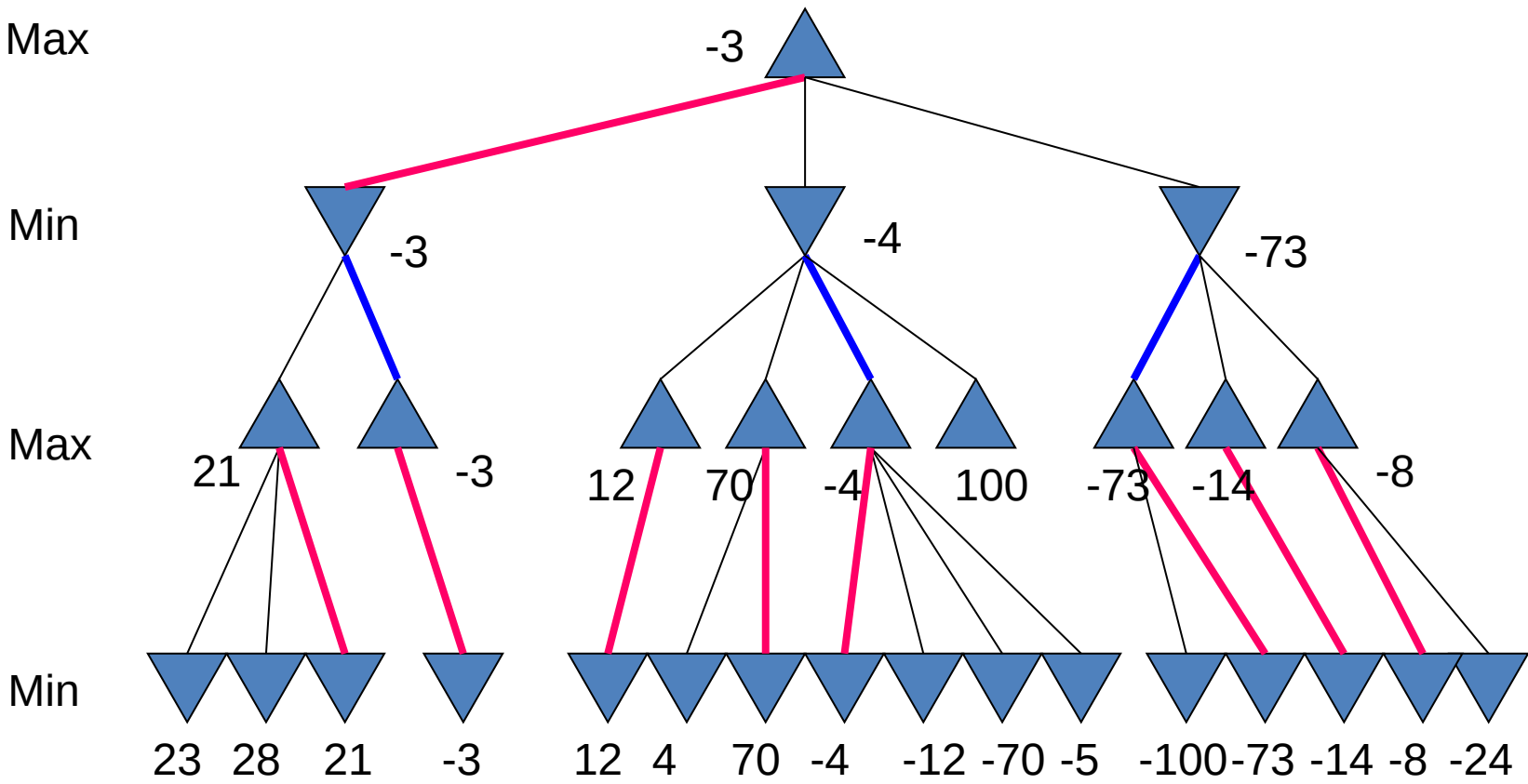
Dr. Abeer Mahmoud
(course coordinator)

Minimax tree



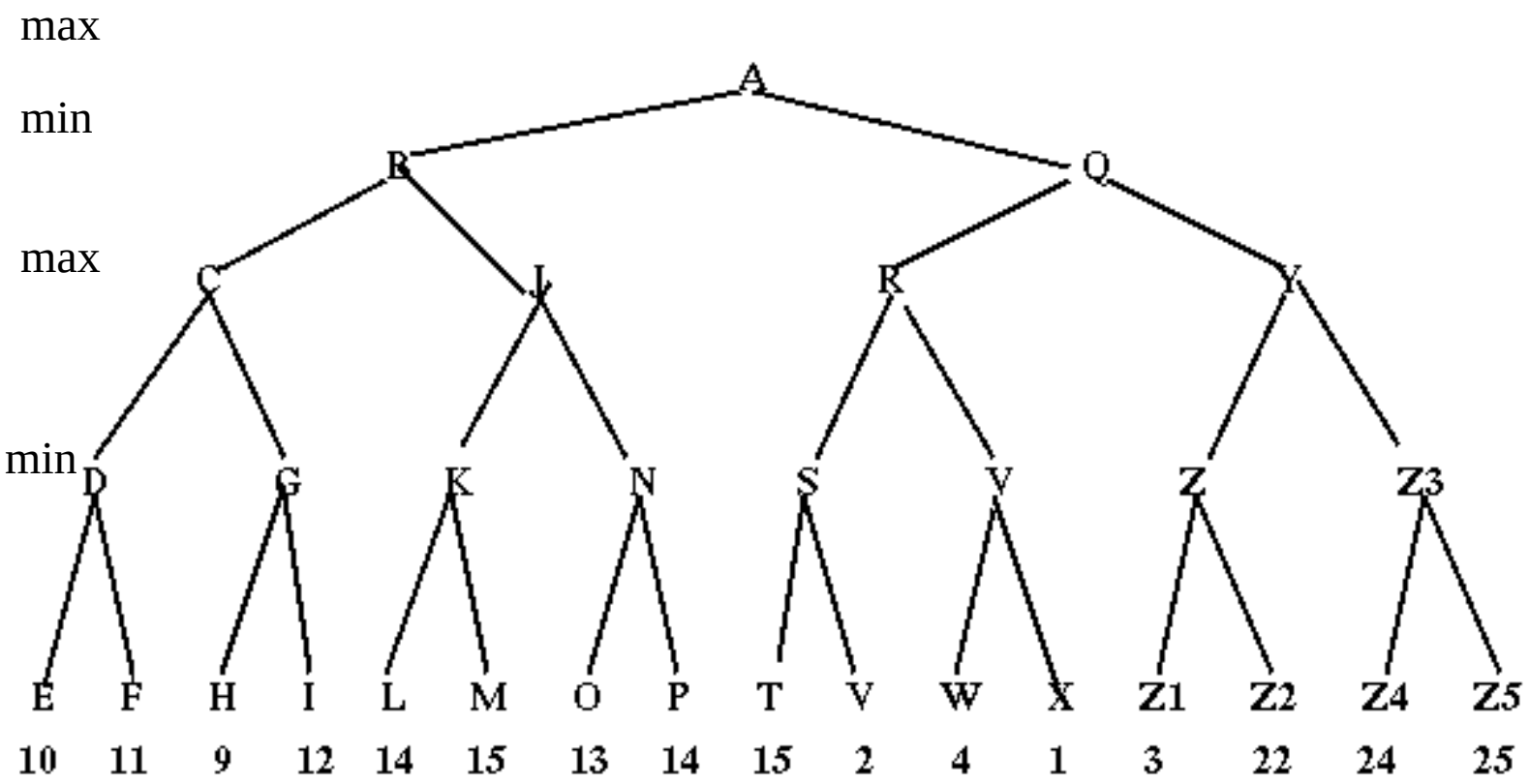
Dr. Abeer Mahmoud
(course coordinator)

Minimax tree



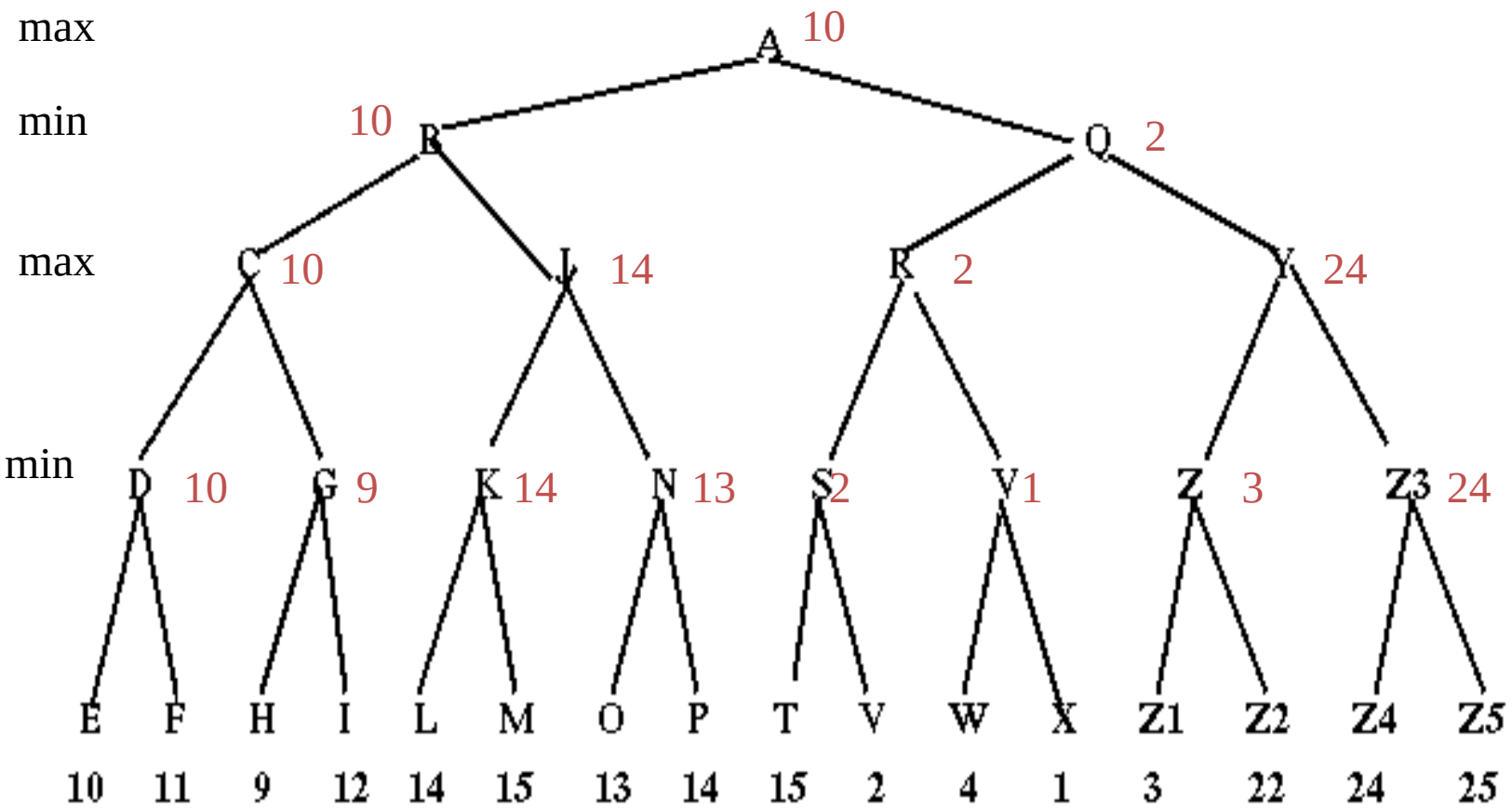
Dr. Abeer Mahmoud
(course coordinator)

Minimax



Dr. Abeer Mahmoud
(course coordinator)

Minimax



Dr. Abeer Mahmoud
(course coordinator)

MinMax Analysis

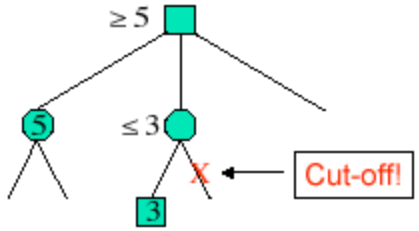
- Time Complexity: $O(b^d)$
- Space Complexity: $O(b*d)$
- Optimality: Yes

Problem: Game → *Resources Limited!*

- Time to make an action is limited

- Can we do better ? Yes !
- How ? Cutting useless branches !

Some nodes in the search can be *proven* to be irrelevent to the outcome of the search



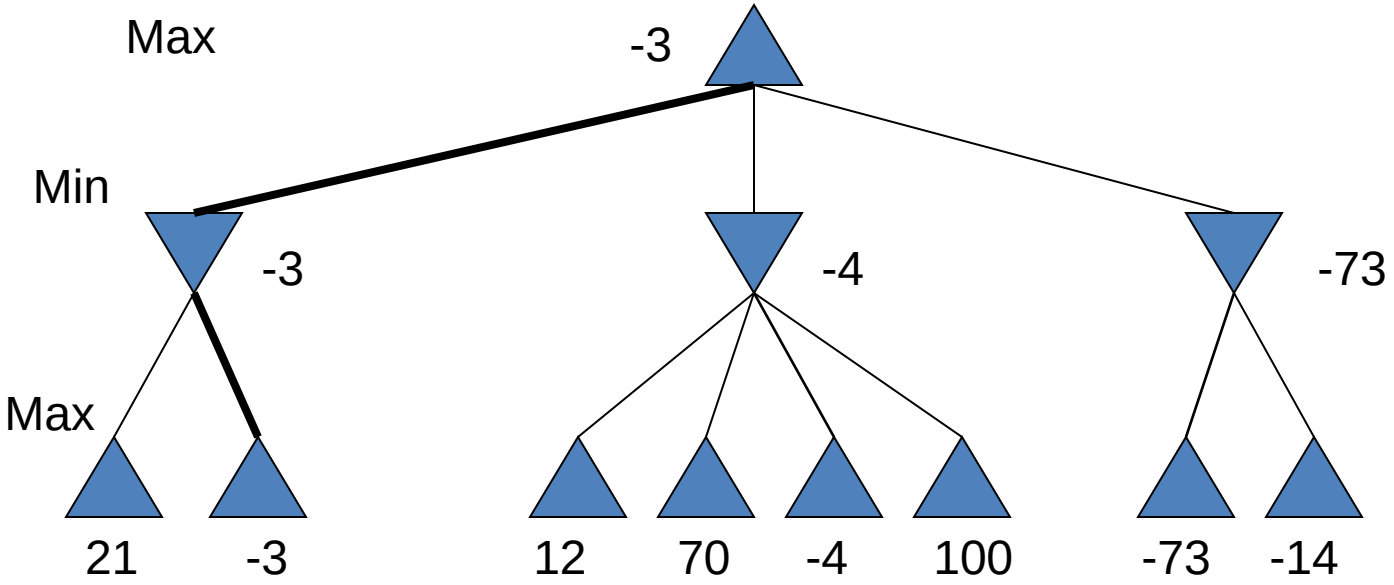
Dr. Abeer Mahmoud
(course coordinator)

α Cuts

- If the current max value is greater than the successor's min value, don't explore that min subtree any more

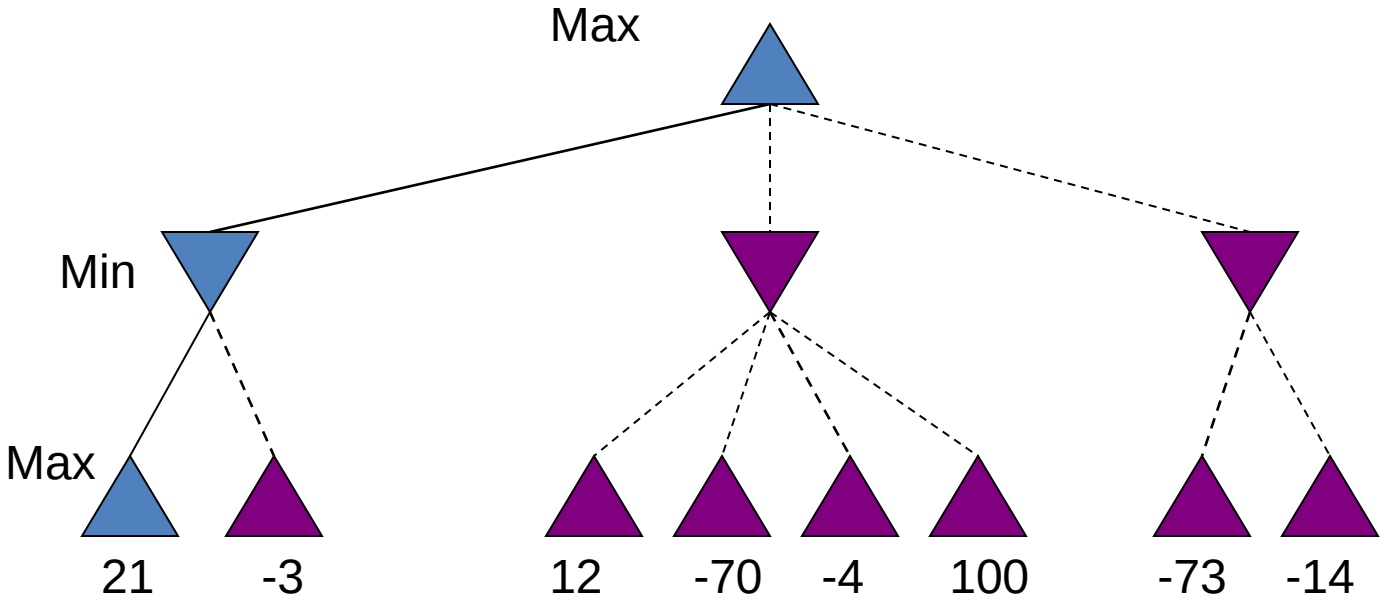
Dr. Abeer Mahmoud
(course coordinator)

α Cut example



Dr. Abeer Mahmoud
(course coordinator)

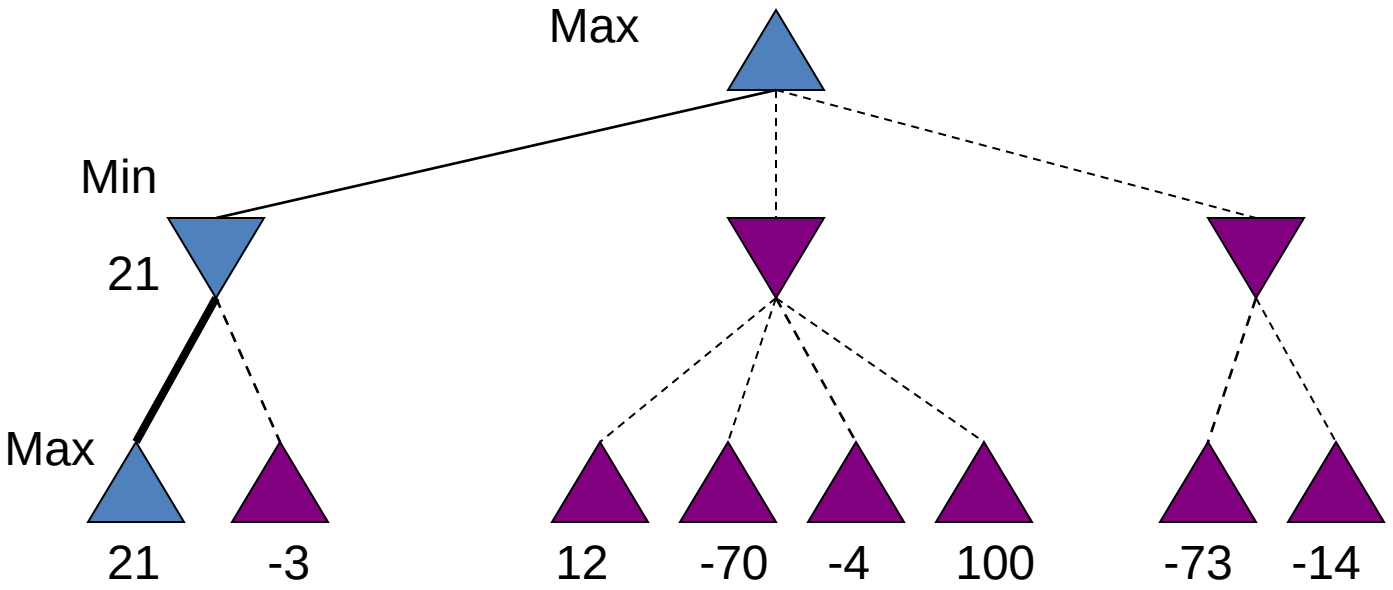
α Cut example



- Depth first search along path 1

Dr. Abeer Mahmoud
(course coordinator)

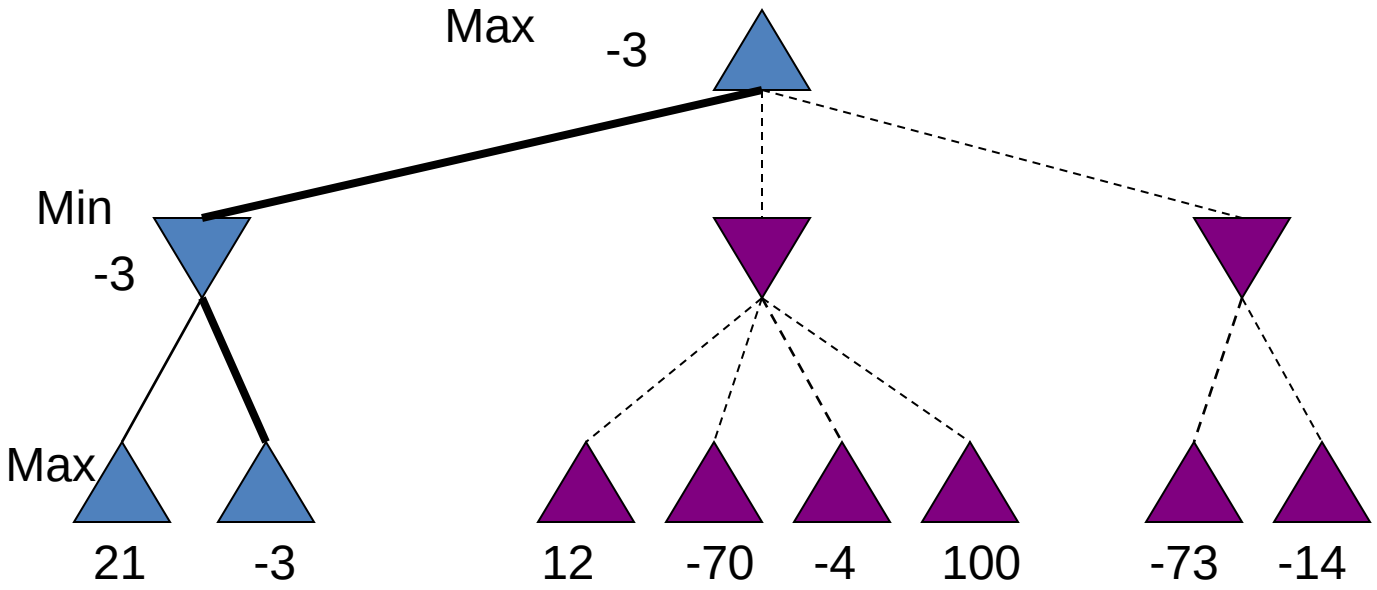
α Cut example



- 21 is minimum so far (second level)
- Can't evaluate yet at top level

Dr. Abeer Mahmoud
(course coordinator)

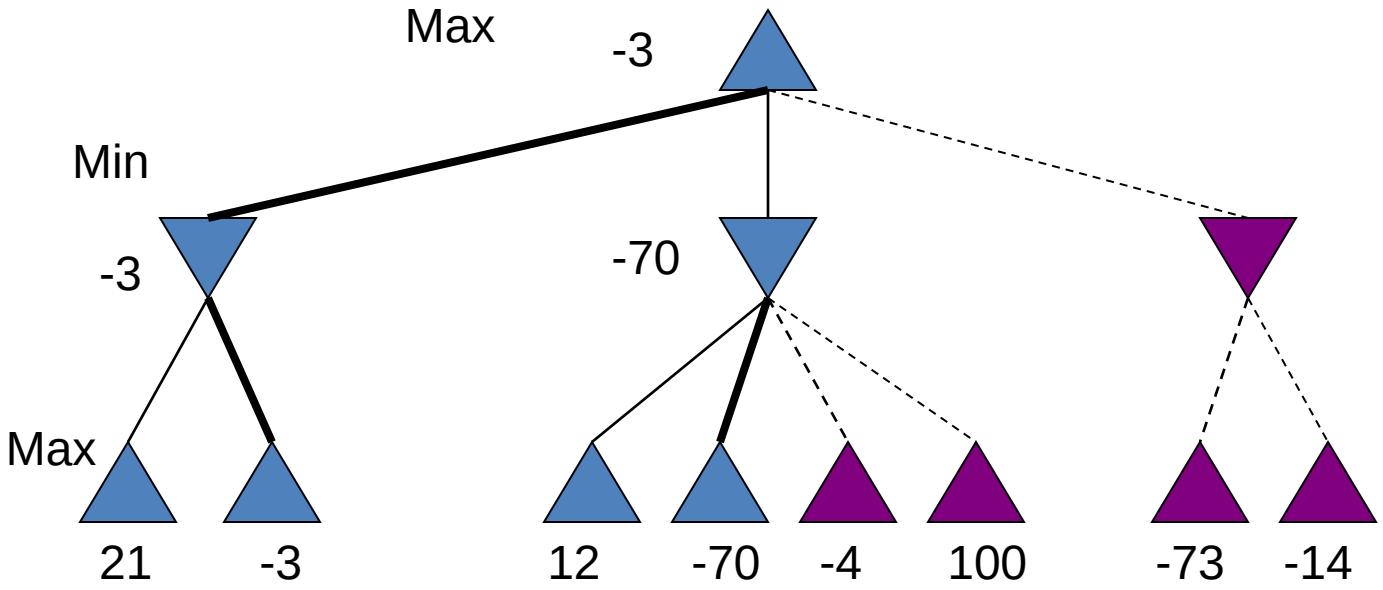
α Cut example



- -3 is minimum so far (second level)
- -3 is maximum so far (top level)

Dr. Abeer Mahmoud
(course coordinator)

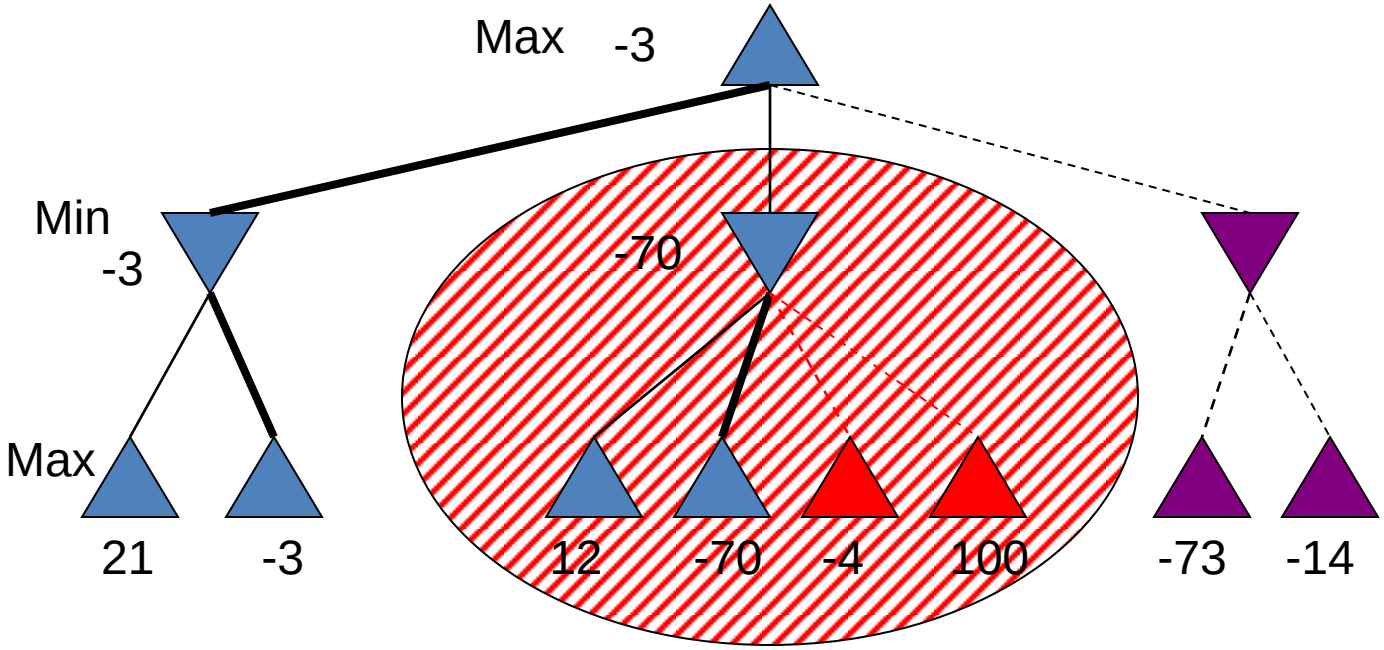
α Cut example



- -70 is now minimum so far (second level)
- -3 is still maximum (can't use second node yet)

Dr. Abeer Mahmoud
(course coordinator)

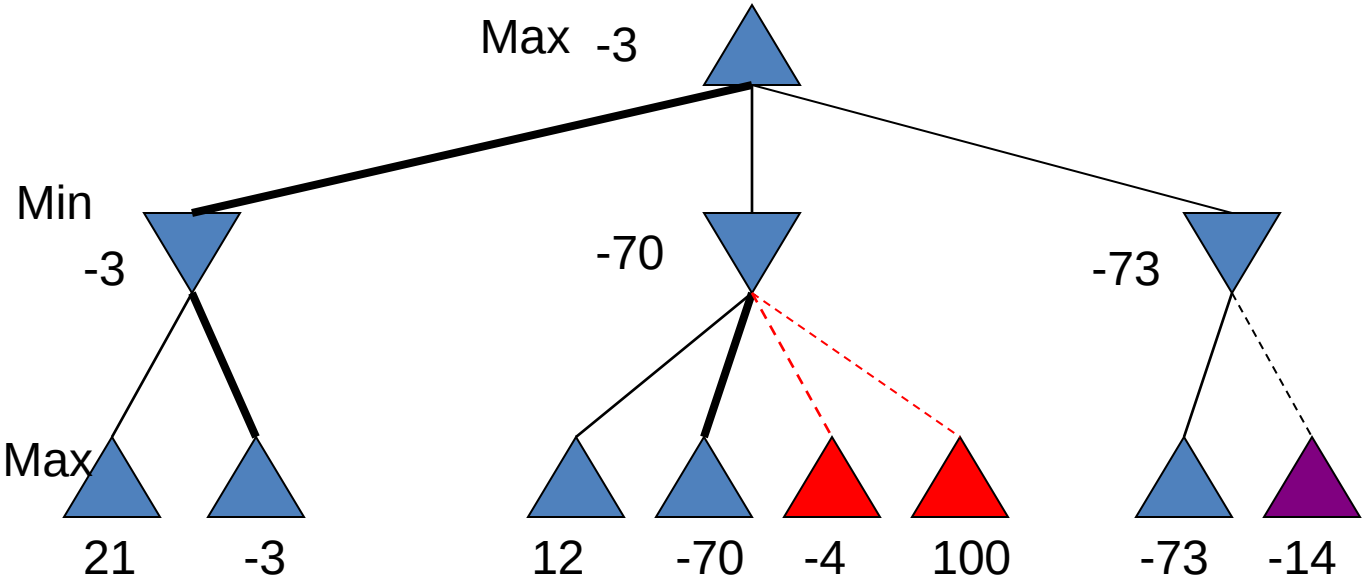
α Cut example



- Since second level node will never be > -70, it will never be chosen by the previous level
- We can stop exploring that node

Dr. Abeer Mahmoud
(course coordinator)

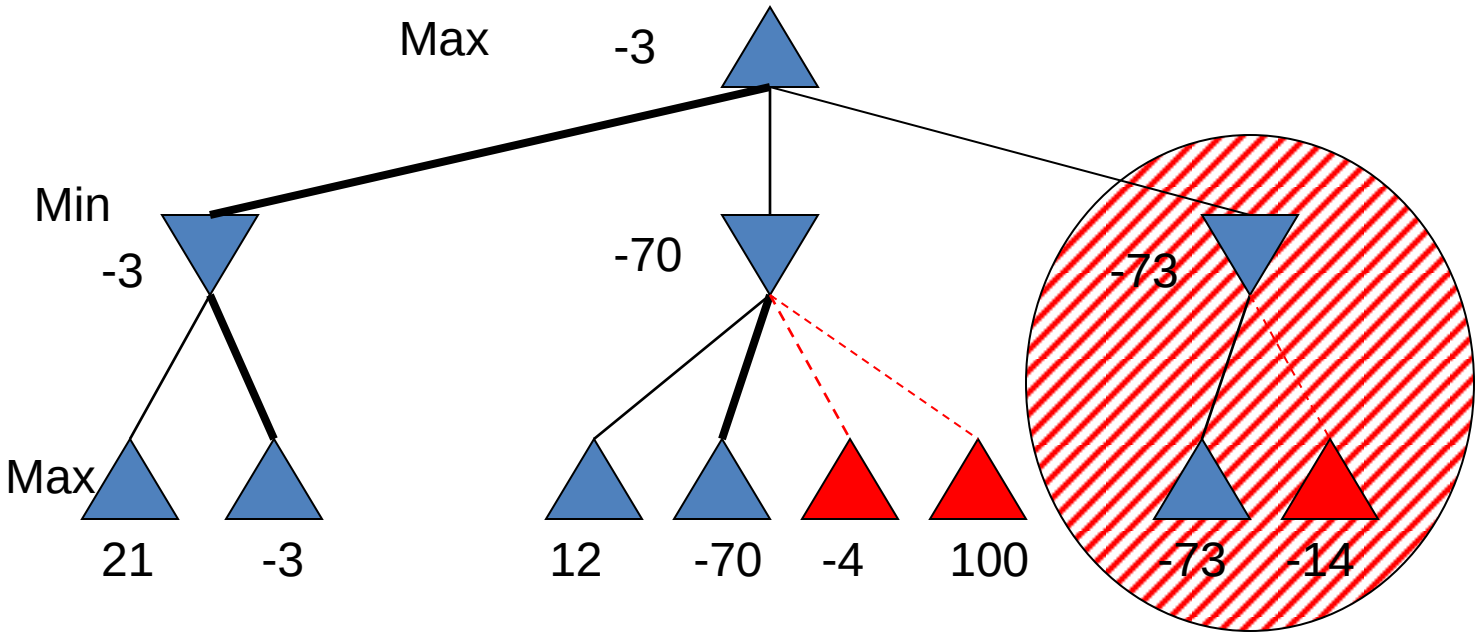
α Cut example



- Evaluation at second level is -73

Dr. Abeer Mahmoud
(course coordinator)

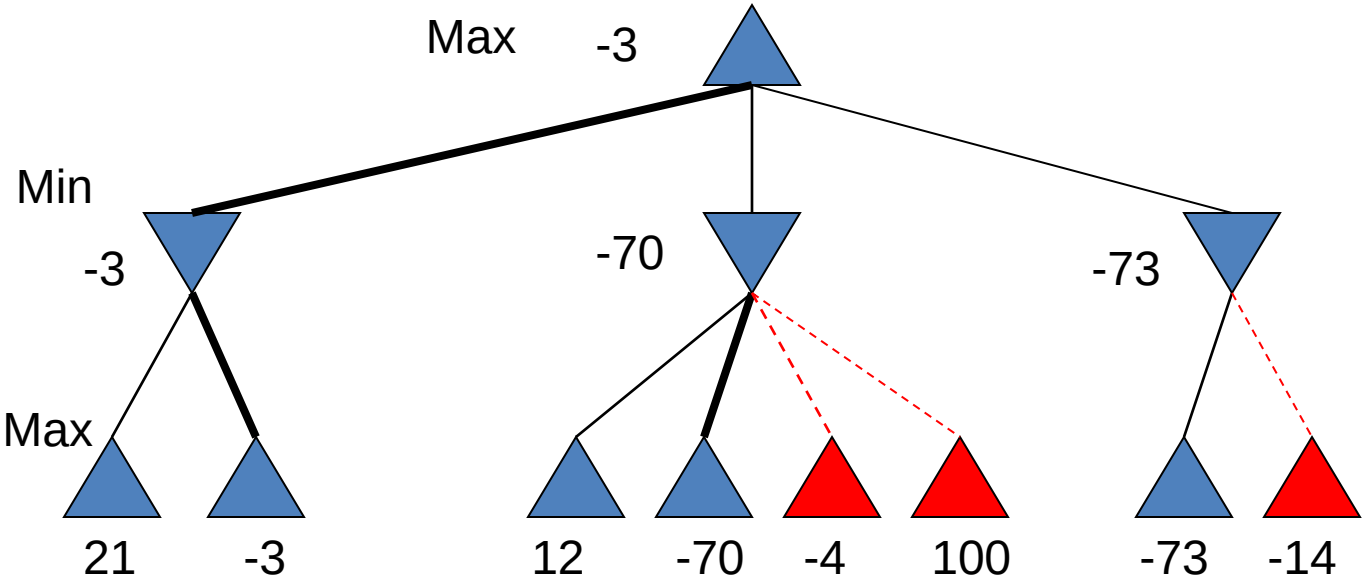
α Cut example



- Again, can apply α cut since the second level node will never be > -73, and thus will never be chosen by the previous level

Dr. Abeer Mahmoud
(course coordinator)

α Cut example



- As a result, we evaluated the Max node without evaluating several of the possible paths

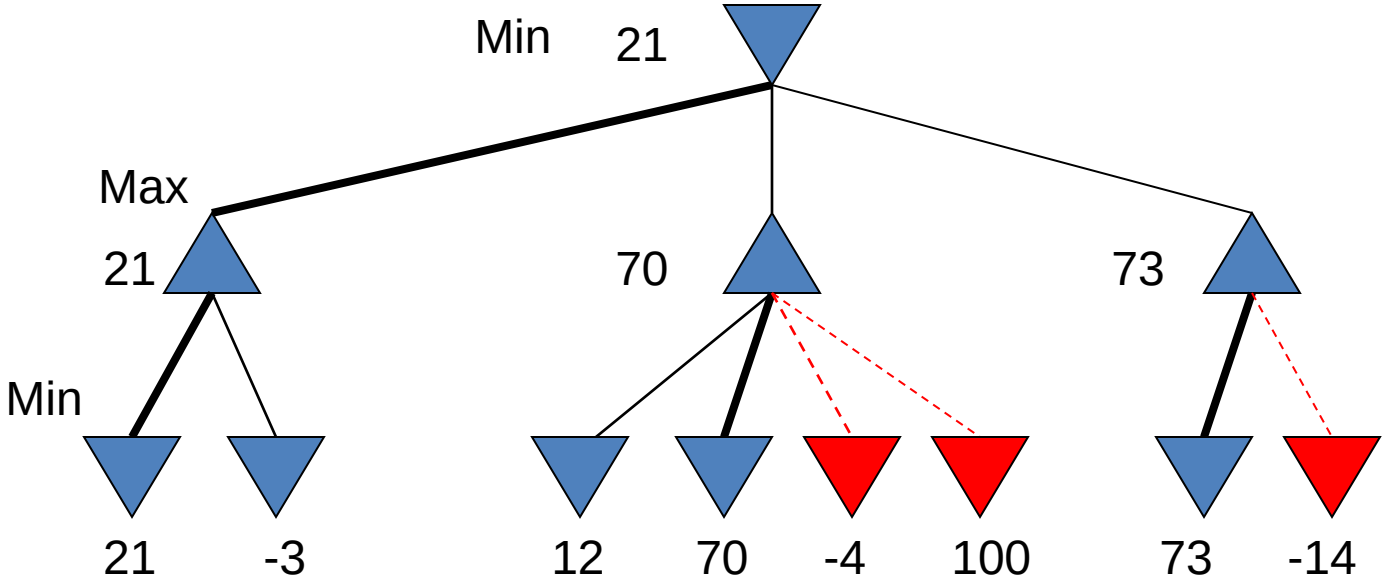
Dr. Abeer Mahmoud
(course coordinator)

β cuts

- Similar idea to α cuts, but the other way around
- If the current minimum is less than the successor's max value, don't look down that max tree any more

Dr. Abeer Mahmoud
(course coordinator)

β Cut example



- Some subtrees at second level already have values > min from previous, so we can stop evaluating them.

Dr. Abeer Mahmoud
(course coordinator)

α-β Pruning

- Pruning by these cuts does not affect final result
  - May allow you to go much deeper in tree
- "Good" ordering of moves can make this pruning much more efficient
  - Evaluating "best" branch first yields better likelihood of pruning later branches
  - Perfect ordering reduces time to $b^{m/2}$
  - i.e. doubles the depth you can search to!

Dr. Abeer Mahmoud
(course coordinator)

α-β Pruning

- Can store information along an entire *path*, not just at most recent levels!
- Keep along the path:
  - α: best MAX value found on this path
    (initialize to most negative utility value)
  - β: best MIN value found on this path
    (initialize to most positive utility value)

Dr. Abeer Mahmoud
(course coordinator)

The Alpha and the Beta

- For a leaf, $\alpha = \beta =$ utility
- At a max node:
  - $\alpha =$ largest child utility found so far
  - $\beta = \beta$ of parent
- At a min node:
  - $\alpha = \alpha$ of parent
  - $\beta =$ smallest child utility found so far
- For any node:
  - $\alpha <=$ utility $<= \beta$
  - "If I had to decide now, it would be..."

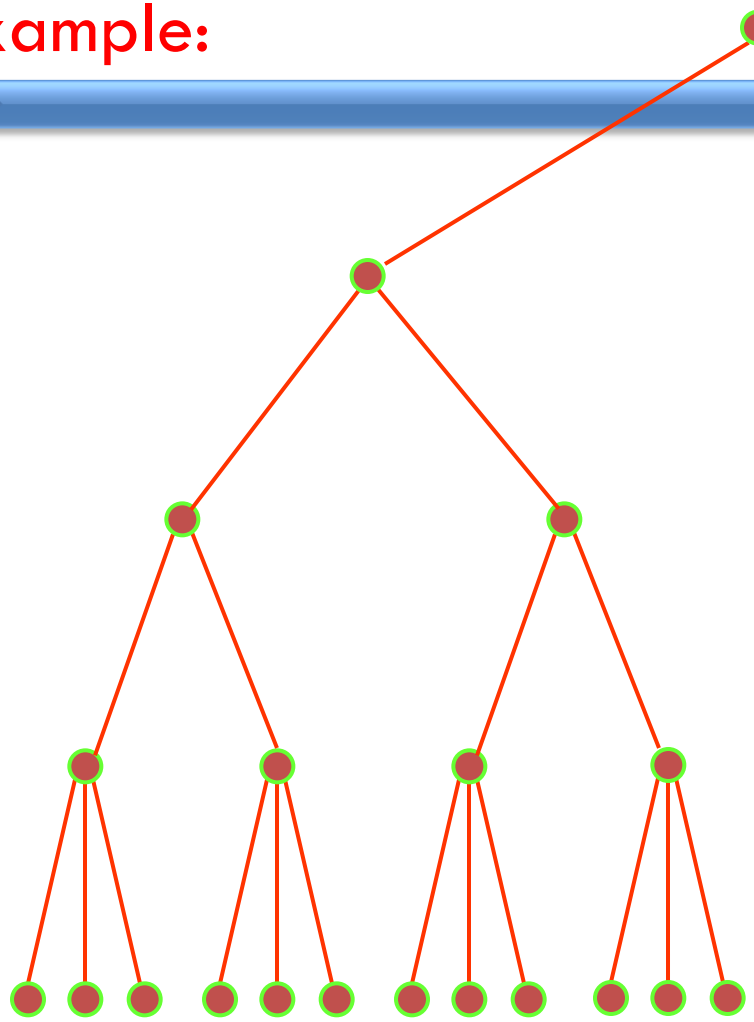
Dr. Abeer Mahmoud
(course coordinator)

Alpha-Beta example

Dr. Abeer Mahmoud
(course coordinator)

The Alpha-Beta Procedure

Example:



Dr. Abeer Mahmoud
(course coordinator)

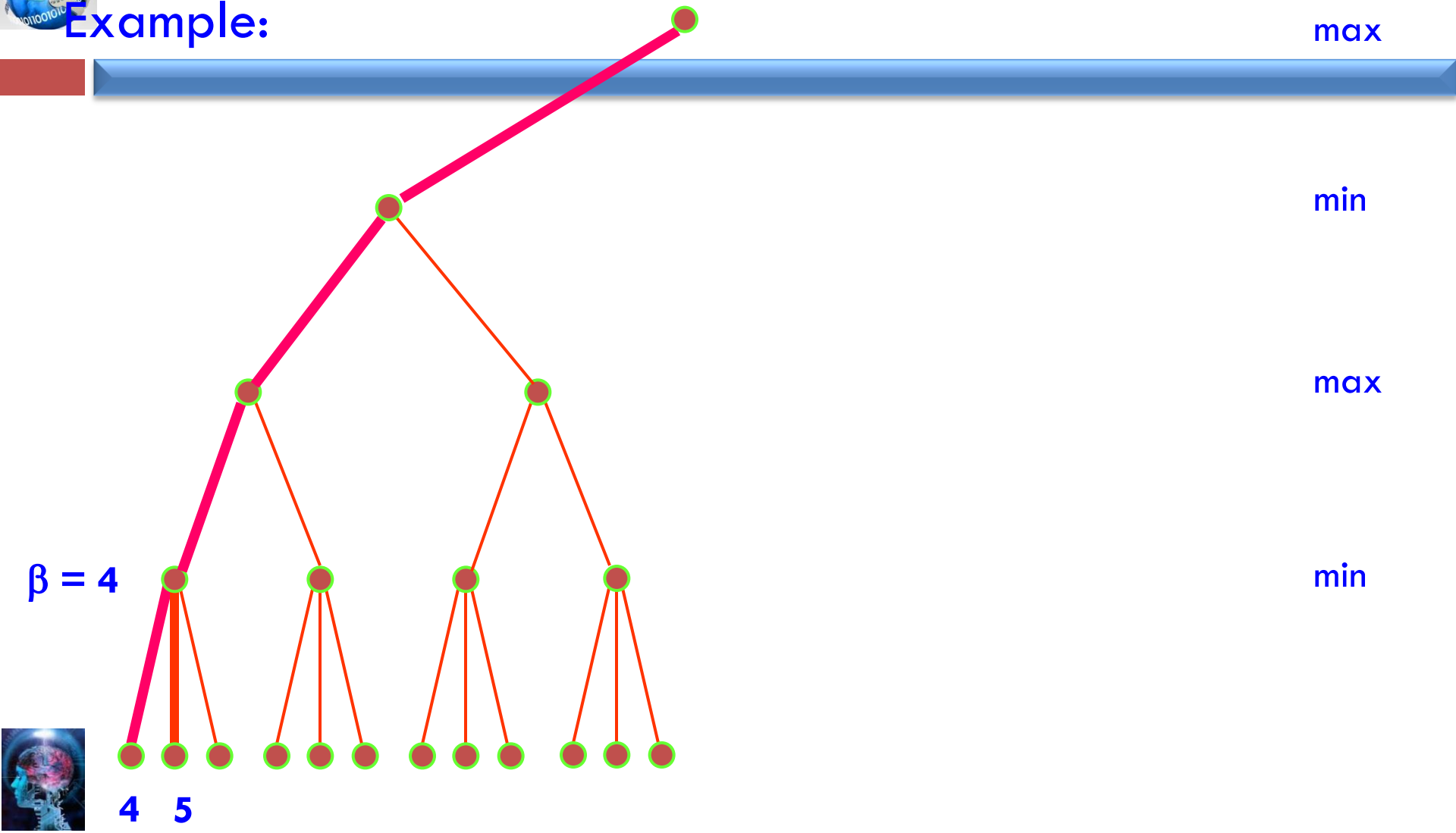
The Alpha-Beta Procedure

Example:

Dr. Abeer Mahmoud
(course coordinator)

The Alpha-Beta Procedure

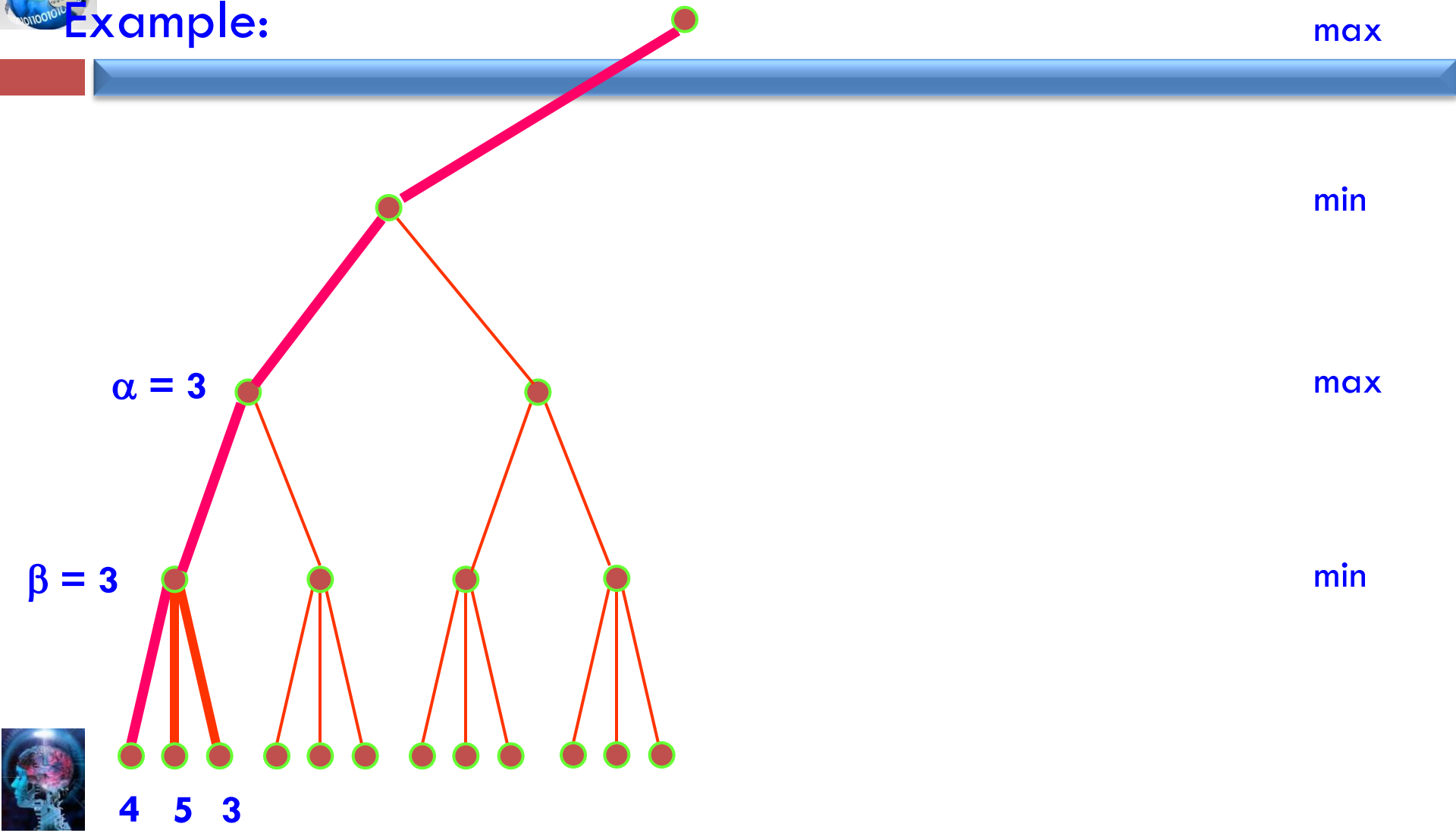
Example:



Dr. Abeer Mahmoud
(course coordinator)

The Alpha-Beta Procedure

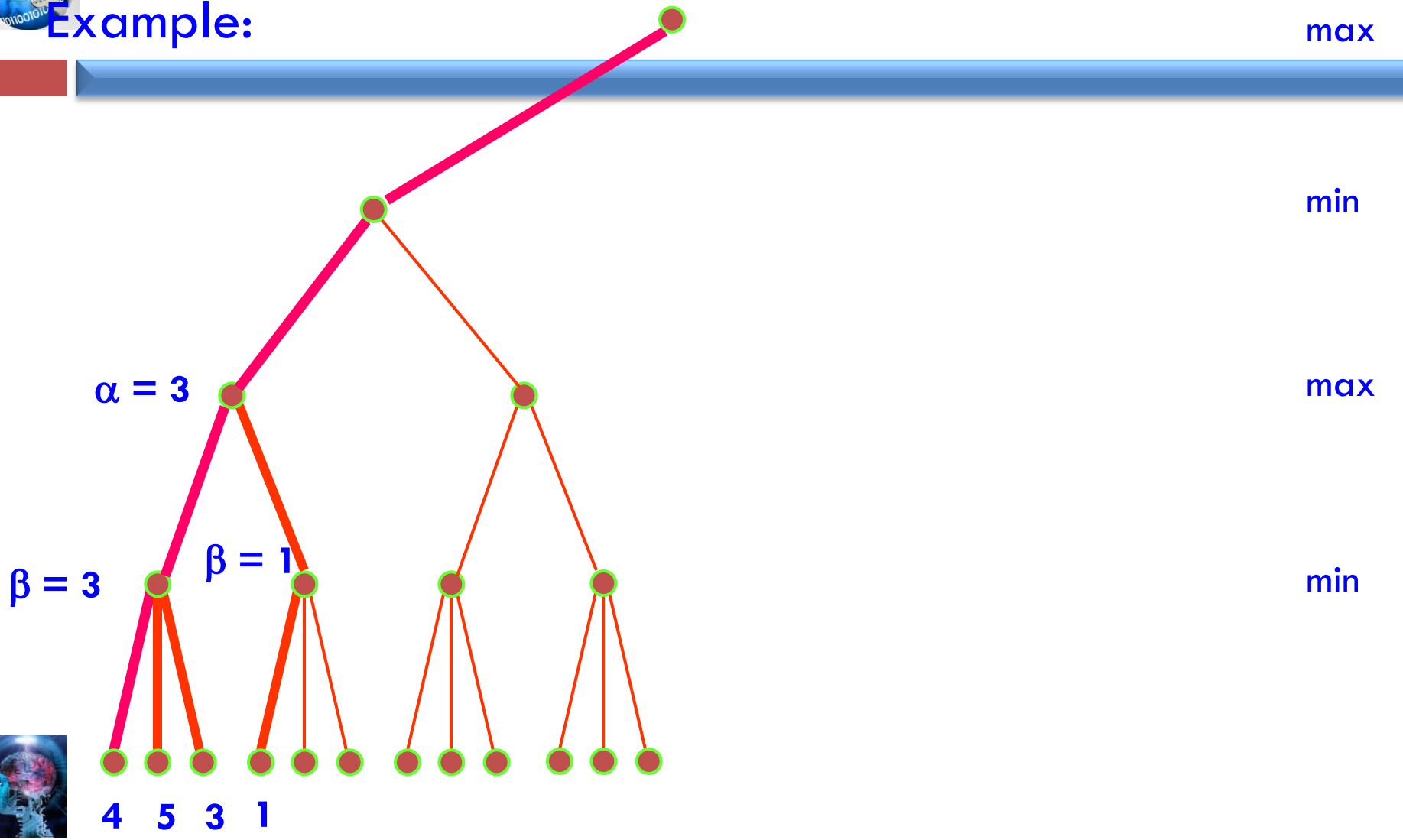
Example:



Dr. Abeer Mahmoud
(course coordinator)

The Alpha-Beta Procedure

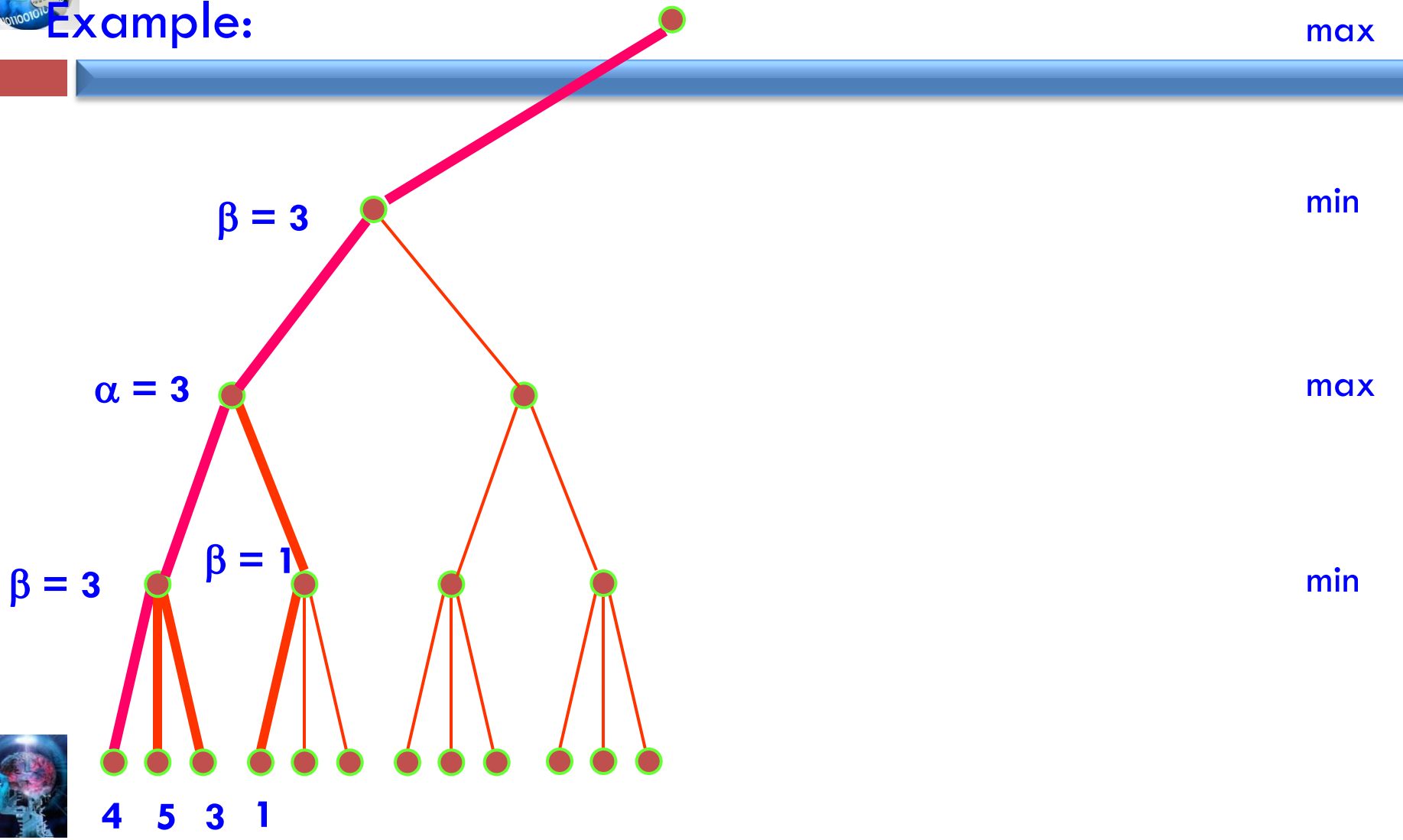
Example:



Dr. Abeer Mahmoud
(course coordinator)

The Alpha-Beta Procedure

Example:



Dr. Abeer Mahmoud
(course coordinator)

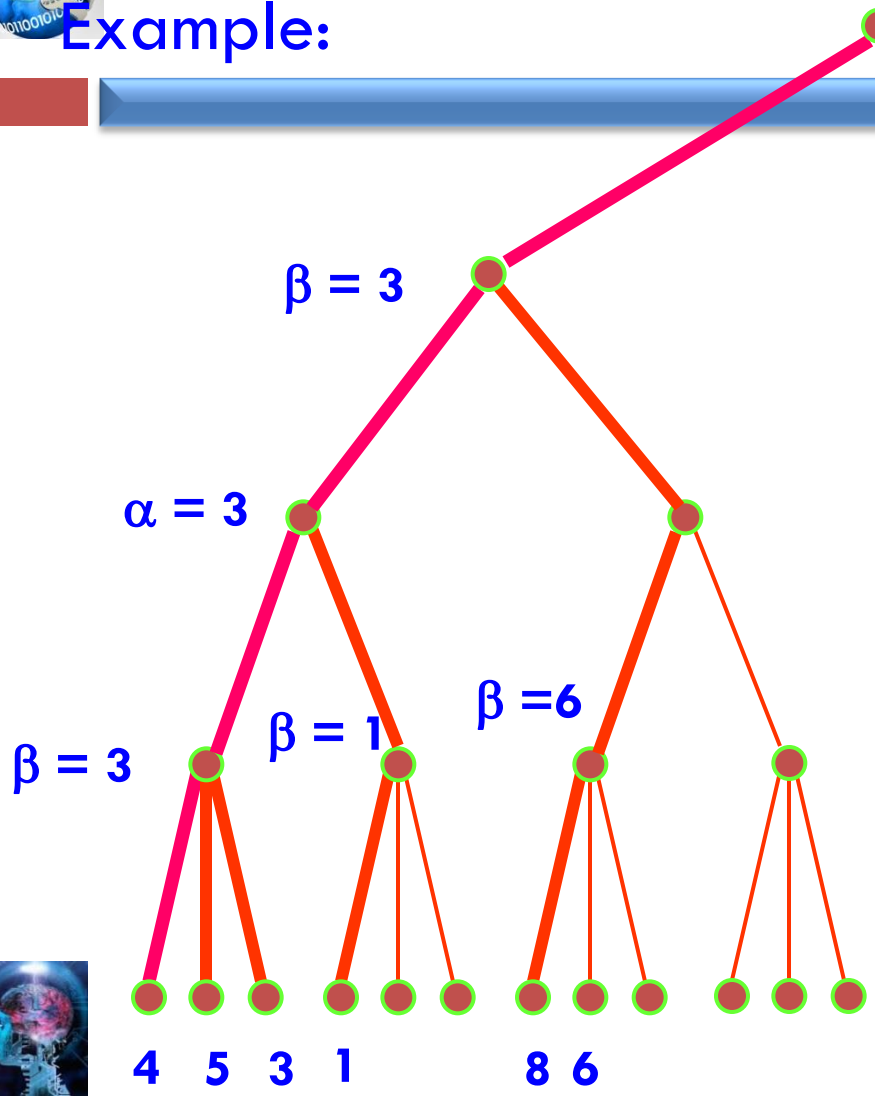
The Alpha-Beta Procedure

Example:

Dr. Abeer Mahmoud
(course coordinator)

The Alpha-Beta Procedure

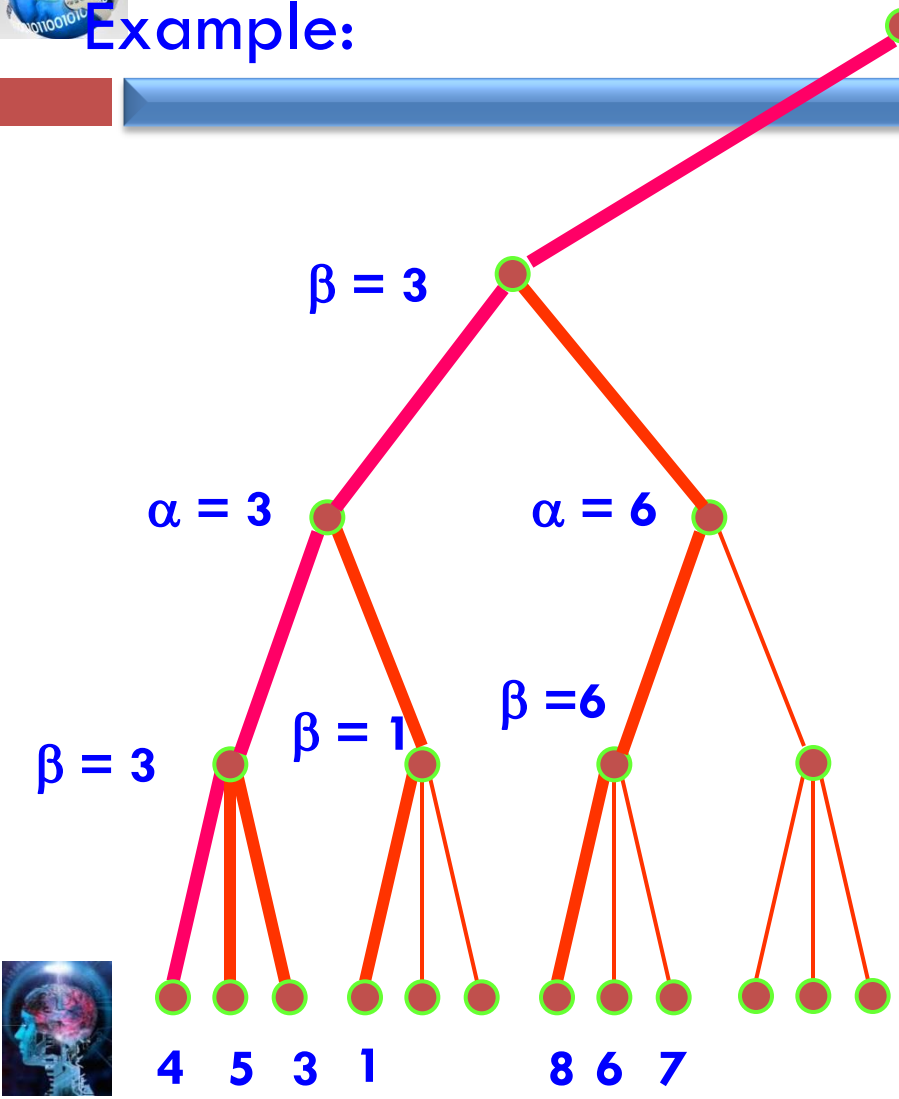
Example:



Dr. Abeer Mahmoud
(course coordinator)

The Alpha-Beta Procedure

Example:



Dr. Abeer Mahmoud
(course coordinator)

Thank you

End of Chapter 5

Dr. Abeer Mahmoud
(course coordinator)