CENG 466
Artificial Intelligence

Lecture 6

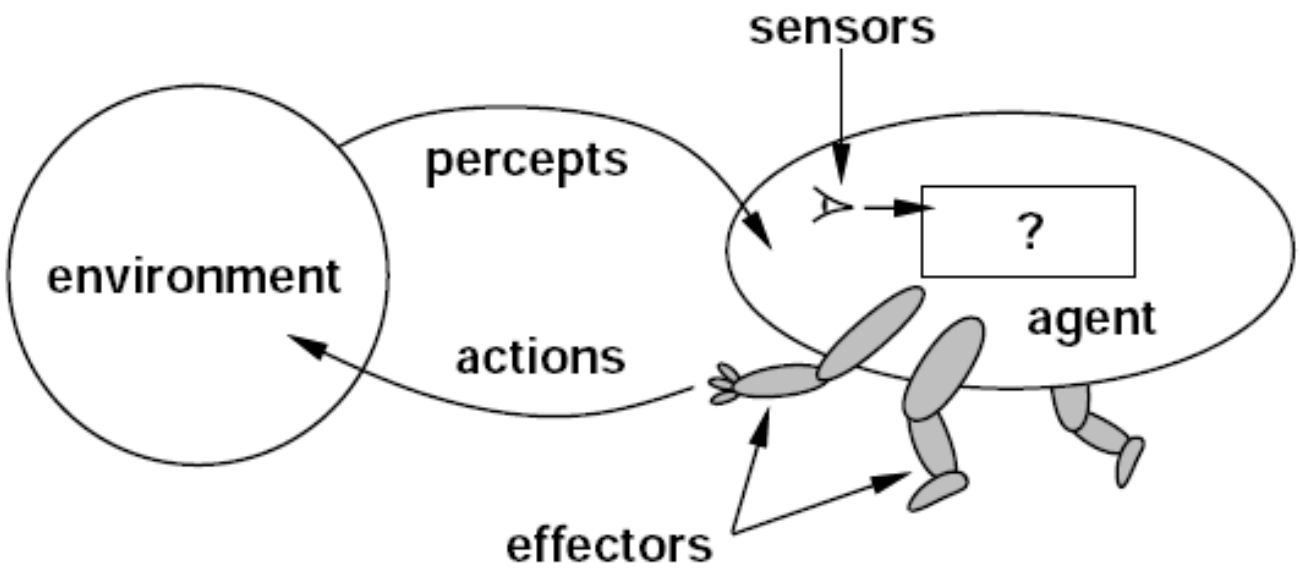
Game Playing

Topics

- Search Categories
- Informed Search Algorithms
- Game Playing
- Game Playing and Min-Max Search
- Perfect Decisions
- Imperfect Decisions
- Evaluation Functions
- Alpha-Beta Pruning
- Games with Chance Element

Intelligent Agents

- An agent is something that perceives and acts in an environment
- An ideal agent always takes actions that maximizes its performance
- An agent adopts a goal and searches the best path to reach that goal



States and State-Spaces

- State: The set of all information items that describe a system at a given time.
- State space is the set of states that an intelligent agent can be in.
- An action takes the agent from one state to another one.
- State space search is finding a sequence of states starting from the initial state to the goal

Searching

- Assuming that the agent knows:
  - how to define a problem,
  - how to recognize a solution (goal),
- finding a solution is done by a search through the state space.

Search Categories

- Un-informed Searches: If we have no extra information about the problem

  (example: depth first search)

- Informed Searches: If we have extra information about the problem.

  (example: A* search)

Un-informed Searches

- In un-informed searches, the agent knows:
  - The initial state
  - The goal state
- But it does not know if a state is close to the goal or not
- Therefore, these searches are blind searches

Informed Searches

- Informed searches have some extra information about the problem
- At each state, we can estimate how far we are from the goal
- Using this information, we can have better search algorithms

How to Use the Information in Searches

- If the state space of a problem is large, we expand only some of the nodes.
- Choosing the next node to expand is the main difference between the search algorithms.
- An informed search algorithm uses its extra information when it decides which node should be expanded

Game Playing

- Game playing is an application of informed searches
- Game playing can be considered as a graph search problem.
- Each node of the graph is a state in the game.
- The goal nodes are the states when we win the game.

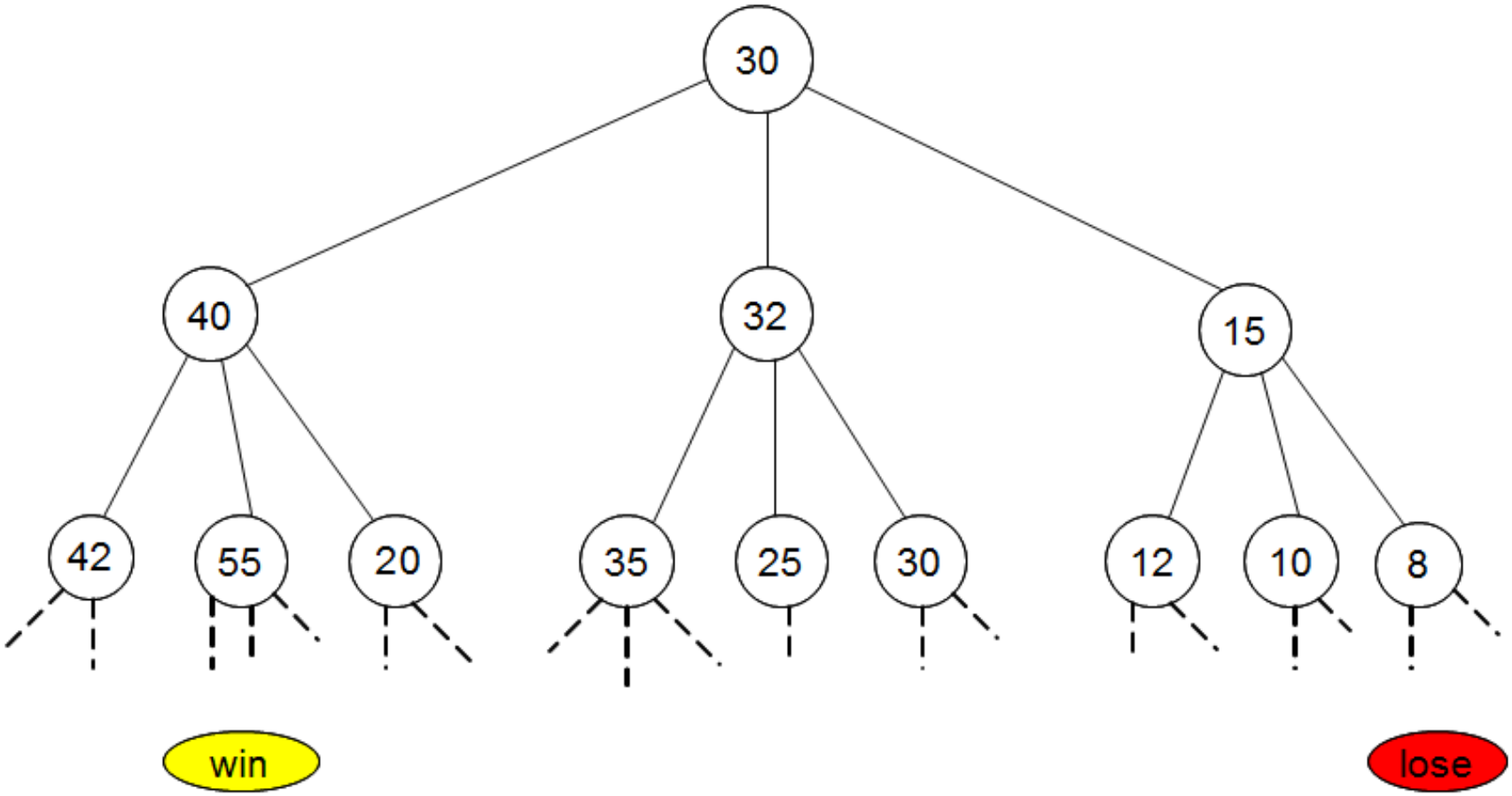
Game Playing Issues

- Game-playing research includes the subjects:
  - How to make the best use of time to reach a goal,
  - How to make a good decision, when reaching optimal decisions is impossible.

The Difference between Games and Search Problems

- In the normal search problems, the search algorithm tries to find the best path to the goal state.
- In a game, the opponents make moves in turns
- Therefore, in one step we want to maximize a value, while in the next step we want to minimize it.
- An algorithm named min-max algorithm is used for game playing.

Min-Max Search



Perfect Decision in Two-Person Games (I)

- A game can be formally defined as a kind of search problem with the following components:
  - **The initial state**, which includes the game position and an indication of whose move it is.
  - **A set of legal moves** that a player can make.
  - A **terminal test**, which determines when the game is over.
  - A **utility function** which gives a numeric value for each state of the game.

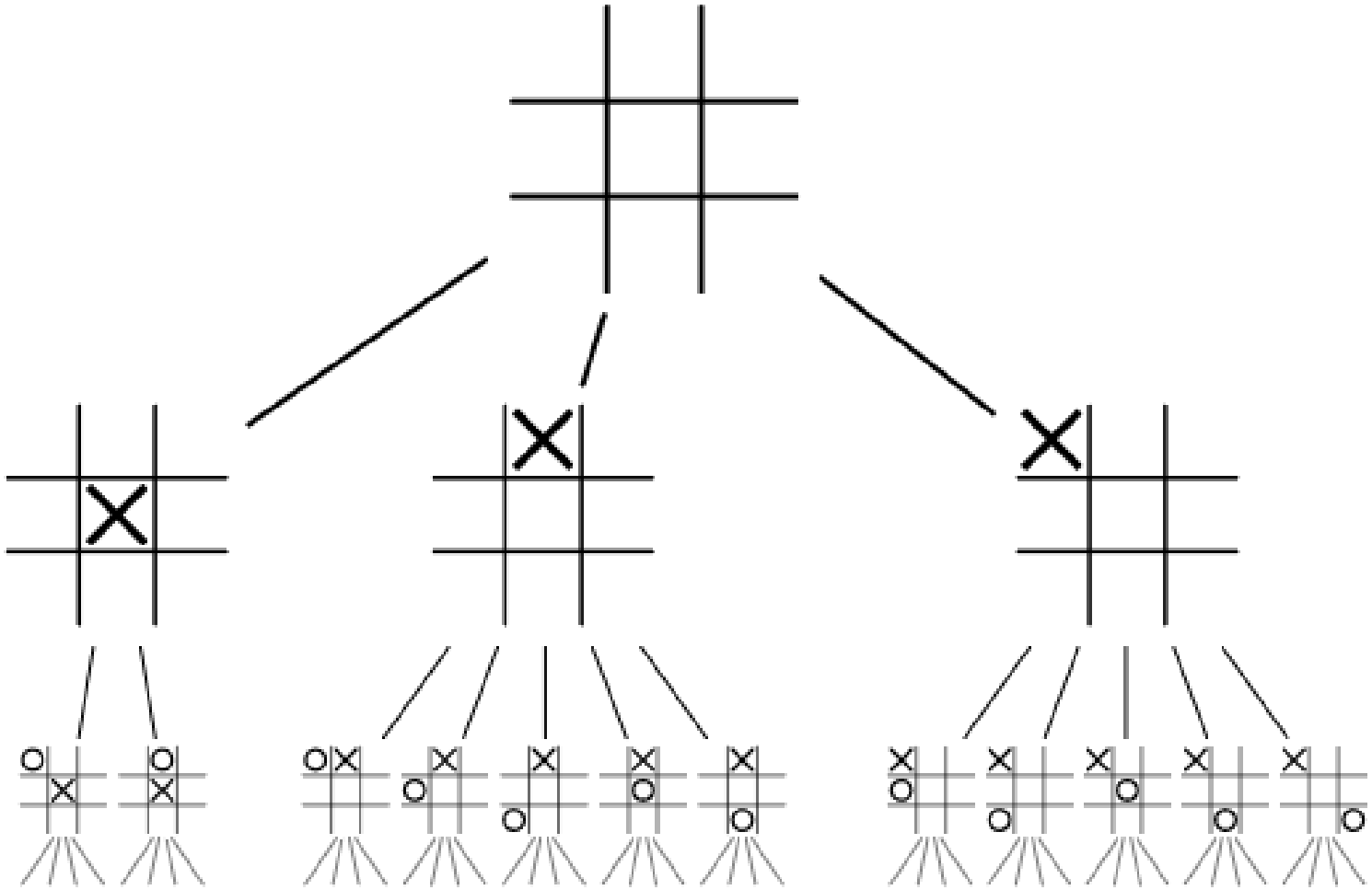
Perfect Decision in Two-Person Games (II)

- In a normal search problem, we follow larger node values which means we are at a better state (close to win) to reach the goal. (MAX step)
- In game search, MIN (the opponent wants to win) is also important.
- Therefore we must find a **strategy** that will lead to a winning terminal state regardless of what MIN does.
- This means we should choose the correct moves for MAX for each possible move of MIN

Perfect Decision in Two-Person Games (III)

- In Perfect Decision using MIN-MAX Search we do:
  - Create the whole tree
  - Assign a value to each node using the utility function
  - Start from the goal node continue backward toward the root, one layer at a time using MAX-MIN values.
  - Finally, at the root point, the sequence to the goal is found
  - This is called the **perfect min-max decision**, because it maximizes the utility under the assumption that the opponent will play perfectly to minimize it.
  - It also assumes we can create the whole tree

Example: Tic-Tac-Toe



Imperfect Decisions

- The min-max algorithm assumes that the program has time to search all the way to the goal (usually not practical)
- Instead of using the utility function, the program should cut off the search earlier and apply a heuristic **evaluation function** to the leaves of the tree.
- We should alter min-max in two ways:
  1. The utility function is replaced by an evaluation function (EVAL)
  2. The terminal test is replaced by a cutoff test CUTOFF-TEST.

Evaluation functions

- An evaluation function returns an estimate of the value of a state (position).
- This value shows how far we are from a win or lose state.

Example: Evaluating Chess Positions

- Chess players can judge the winning chances of each side.
- For example, MATERIAL VALUE gives an approximate value for each piece:
  - each pawn is worth 1,
  - a knight is worth 3,
  - a bishop is worth 2,
  - a castle 5,
  - a queen 9.
- All other things being equal, a 3-point advantage is sufficient for near-certain victory.

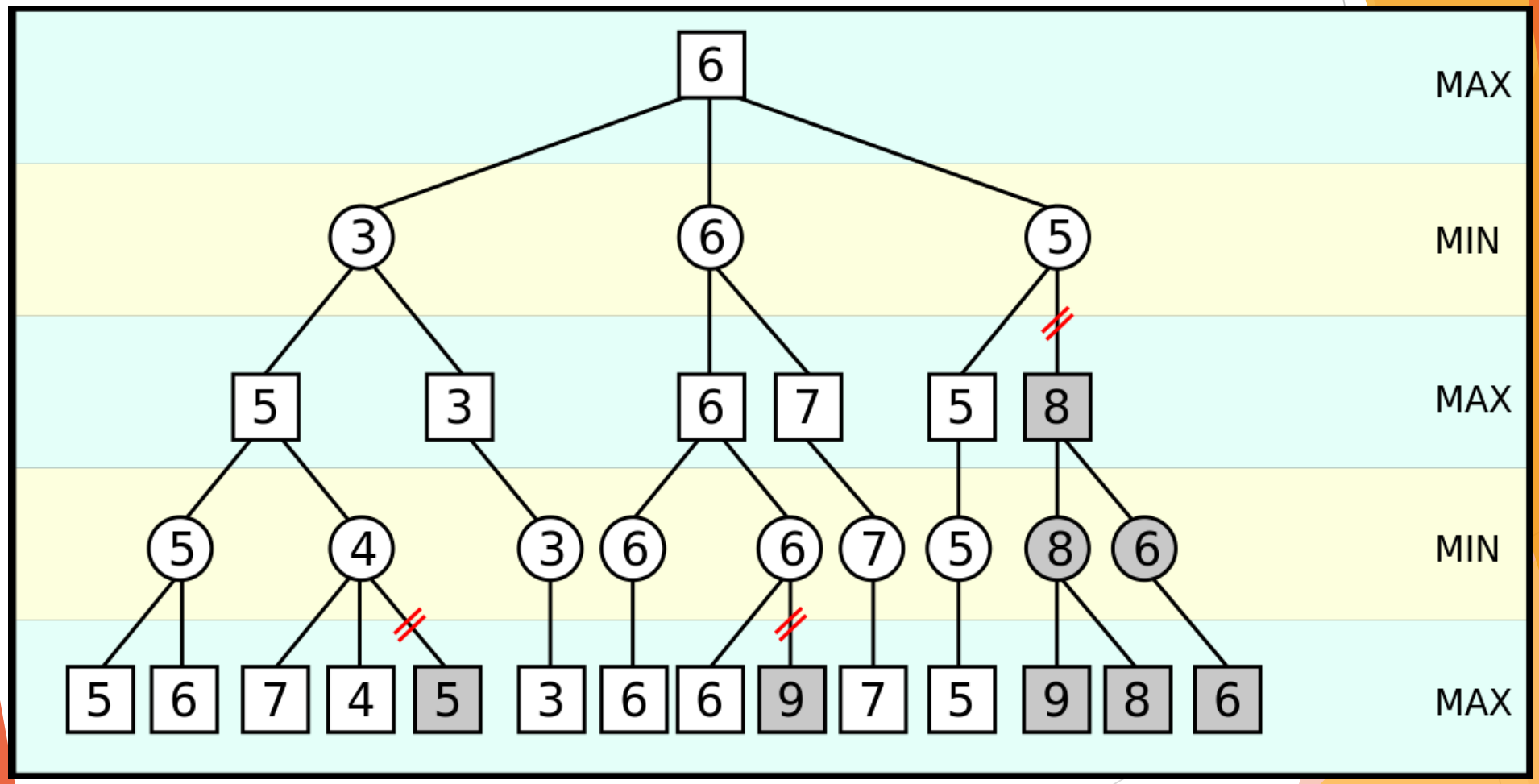
Cutting Off Search

- The most common approach to control the search is to set a fixed depth limit.
- The depth is chosen so that the amount of time used will not exceed what the rules of the game allow.
- A slightly better approach is to apply iterative deepening.
- When time runs out, the program returns the move selected by the deepest completed search.

Alpha-Beta Pruning

- Sometimes searching all branches of a tree when we have time limit is not possible.
- In these cases we can cut some of the branches.
- The remaining branches are searched in deeper levels.
- One of the algorithms to cut (prune) branches of a tree is **Alpha-Beta Pruning**

Alpha-Beta Pruning Example



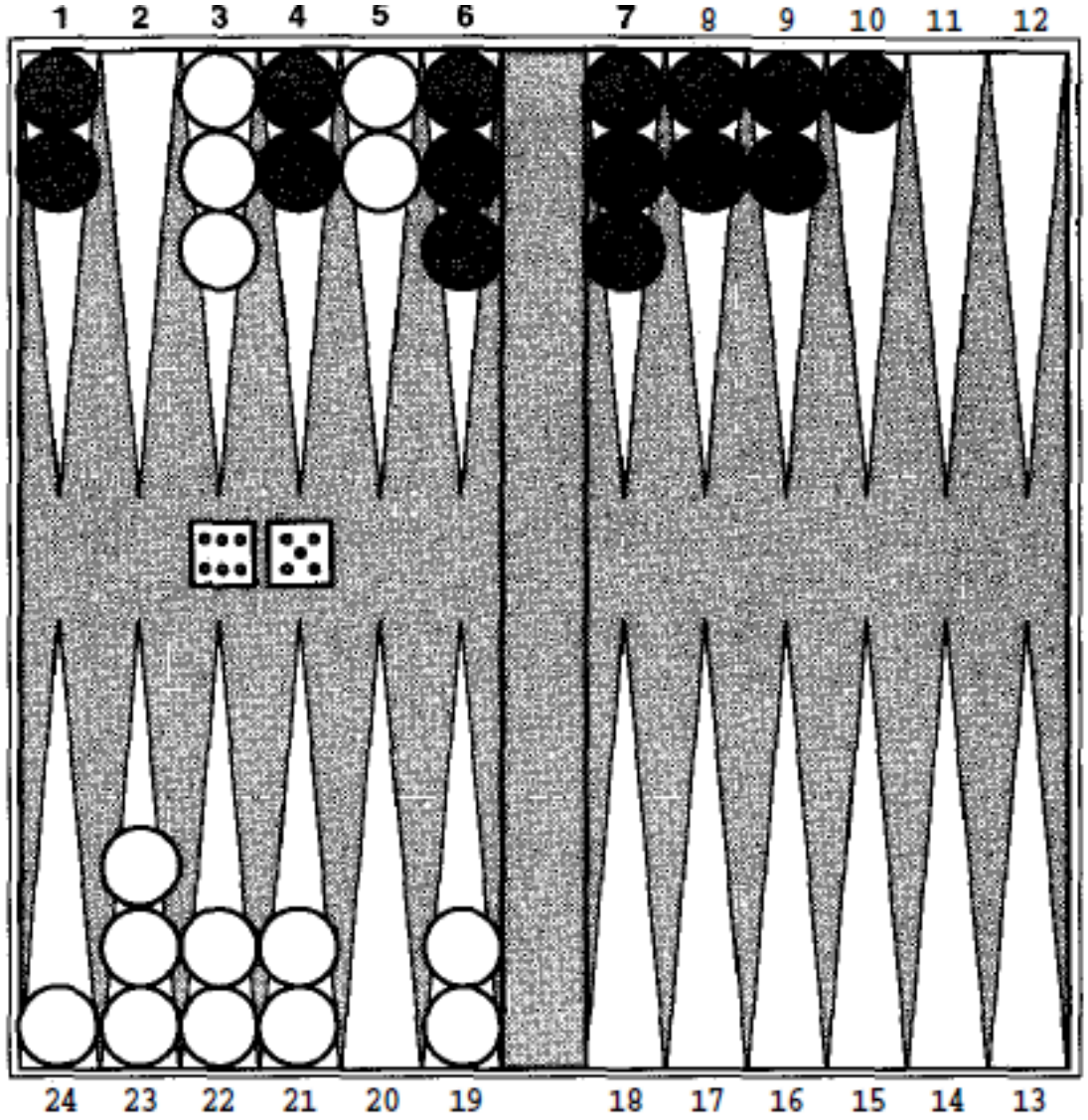
Games that Include an Element of Chance (I)

- Some games combine luck and skill.
- Backgammon is a typical example.
- Dice are rolled at the beginning of a player's turn to determine the set of legal moves that is available to the player.

Games that Include an Element of Chance (II)

- After rolling the dice, the player knows his/her possible moves
- The player does not know what the other player will roll (and his legal moves).
- That means he cannot construct a complete game tree.
- A game tree in these games must include **chance nodes** in addition to MAX and MIN nodes.

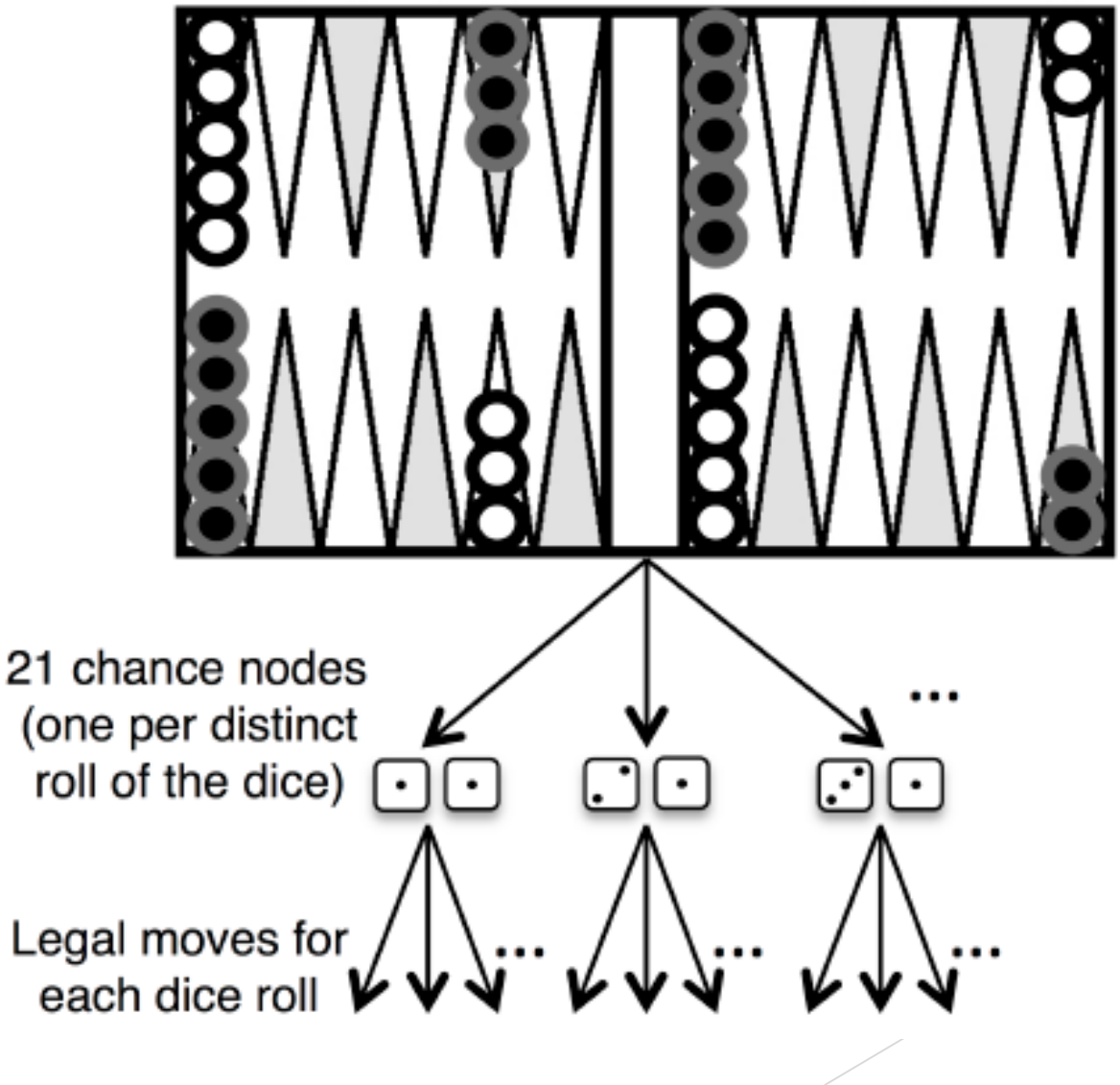
Example



Chance Nodes

- In the backgammon example, white has rolled a 6-5, and has four possible moves.
- Although white knows what his or her own legal moves are, white does not know what black is going to roll, and thus does not know what black's legal moves will be.
- That means white cannot construct a complete game tree like we saw in chess and Tic-Tac-Toe.
- A game tree in backgammon must include **chance nodes** in addition to MAX and MIN nodes.

Example (I)


21 chance nodes
(one per distinct
roll of the dice)
Legal moves for
each dice roll

Position evaluation in games with chance nodes

- In games with chance, we do not know what actions we will have in the next step.
- Therefore, evaluation function should consider all consequences of a movement.
- The evaluation function can consider the probability of winning from a position

Example: Backgammon

Position Evaluation in Backgammon Example

- The evaluation depends on the probability of rolling each number.
- If black plays with its farthest piece and takes the single piece of the white player, it should consider the probability of (1,1) or (1,6), or (6,1)

{1/36+1/36+1/36}

in evaluating the position after that move

Questions?