CSC384: Intro to Artificial Intelligence
Game Tree Search I

- Readings
  - Chapter 6.1, 6.2, 6.3, 6.6
- Announcements:
  - Sample Questions for Test1 posted on the web!
  - T3: Fri Sep 29 10-11am SS2127
  - T4: Fri Sep 29 4-5pm BA3012

Generalizing Search Problems

- So far: our search problems have assumed agent has complete control of environment
  - state does not change unless the agent (robot) changes it.
    - makes a straight path to goal state feasible.
- Assumption not always reasonable
  - stochastic environment (e.g., the weather, traffic accidents).
  - other agents whose interests conflict with yours

Generalizing Search Problems

- In these cases, we need to generalize our view of search to handle state changes that are not in the control of the agent.
- One generalization yields game tree search
  - agent and some other agents.
  - The other agents are acting to maximize their profits
    - this might not have a positive effect on your profits.

Two-person Zero-Sum Games

- Two-person, zero-sum games
  - chess, checkers, tic-tac-toe, backgammon, go, Doom, “find the last parking space”
  - Your winning means that your opponent looses, and vice-versa.
  - Zero-sum means the sum of your and your opponent’s payoff is zero---any thing you gain come at your opponent’s cost (and vice-versa). Key insight:
  - how you act depends on how the other agent acts (or how you think they will act)
    - and vice versa (if your opponent is a rational player)

More General Games

- What makes something a game?
  - there are two (or more) agents influencing state change
  - each agent has their own interests
    - e.g., goal states are different; or we assign different values to different paths/states
  - Each agent tries to alter the state so as to best benefit itself.

More General Games

- What makes games hard?
  - how you should play depends on how you think the other person will play; but how they play depends on how they think you will play; so how you should play depends on how you think they think you will play; but how they play should depend on how they think you think they think you will play; …

More General Games

- Zero-sum games are "fully competitive"
  - if one player wins, the other player loses
  - e.g., the amount of money I win (lose) at poker is the amount of money you lose (win)
- More general games can be "cooperative"
  - some outcomes are preferred by both of us, or at least our values aren't diametrically opposed
- We'll look in detail at zero-sum games
  - but first, some examples of simple zero-sum and cooperative games

Game 1: Rock, Paper Scissors

- Scissors cut paper, paper covers rock, rock smashes scissors
- Represented as a matrix: Player I chooses a row, Player II chooses a column
- Payoff to each player in each cell (Pl.I / Pl.II)
- 1: win, 0: tie, –1: loss
  - so it's zero-sum

Player II

Player I	R	P	S
R	0/0	-1/1	1/-1
P	1/-1	0/0	-1/1
S	-1/1	1/-1	0/0

Game 2: Prisoner's Dilemma

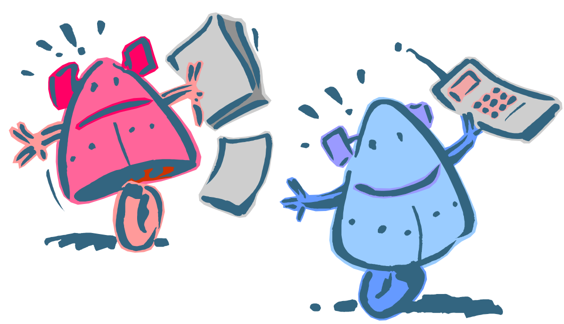
- Two prisoner's in separate cells, DA doesn't have enough evidence to convict them
- If one confesses, other doesn't:
  - confessor goes free
  - other sentenced to 4 years
- If both confess (both defect)
  - both sentenced to 3 years
- Neither confess (both cooperate)
  - sentenced to 1 year on minor charge
- Payoff: 4 minus sentence

	Coop	Def
Coop	3/3	0/4
Def	4/0	1/1

Game 3: Battlebots

- Two robots: Blue (Craig's), Red (Fahiem's)
  - one cup of coffee, one tea left
  - both C, F prefer coffee (value 10)
  - tea acceptable (value 8)
- Both robot's go for Cof
  - collide and get no payoff
- Both go for tea: same
- One goes for coffee, other for tea:
  - coffee robot gets 10
  - tea robot gets 8

	C	T
C	0/0	10/8
T	8/10	0/0



Two Player Zero Sum Games

- Key point of previous games: what you should do depends on what other guy does
- Previous games are simple "one shot" games
  - single move each
  - in game theory: *strategic or normal form games*
- Many games extend over multiple moves
  - e.g., chess, checkers, etc.
  - in game theory: *extensive form games*
- We'll focus on the extensive form
  - that's where the computational questions emerge

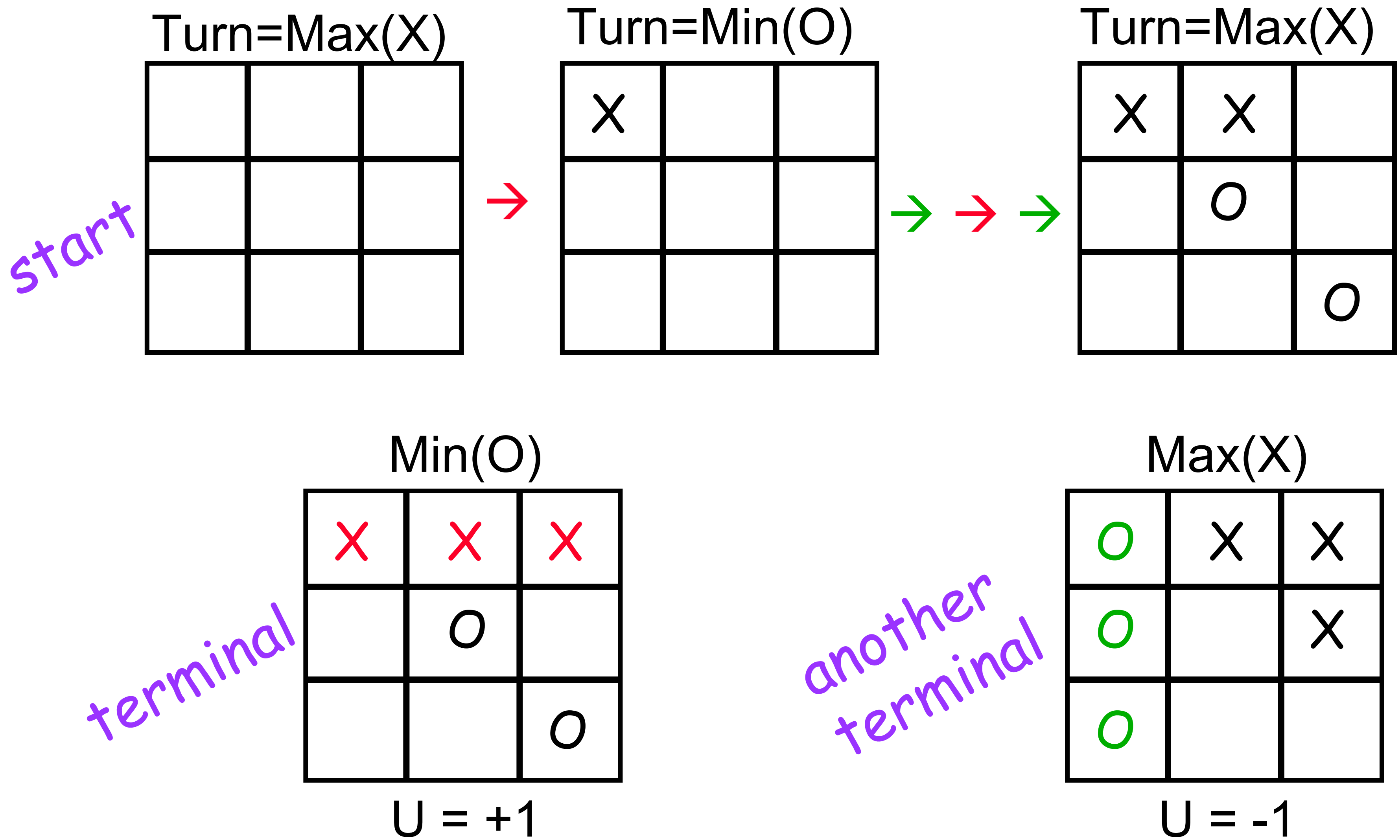
Two-Player, Zero-Sum Game: Defn

- Two *players* A (Max) and B (Min)
- set of *positions* P (states of the game)
- a *starting position* $s \in P$ (where game begins)
- *terminal positions* $T \subseteq P$ (where game can end)
- set of directed edges E_A between states (A's *moves*)
- set of directed edges E_B between states (B's *moves*)
- *utility* or *payoff function* $U : T \rightarrow \mathbb{R}$ (how good is each terminal state for player A)
  - why don't we need a utility function for B?

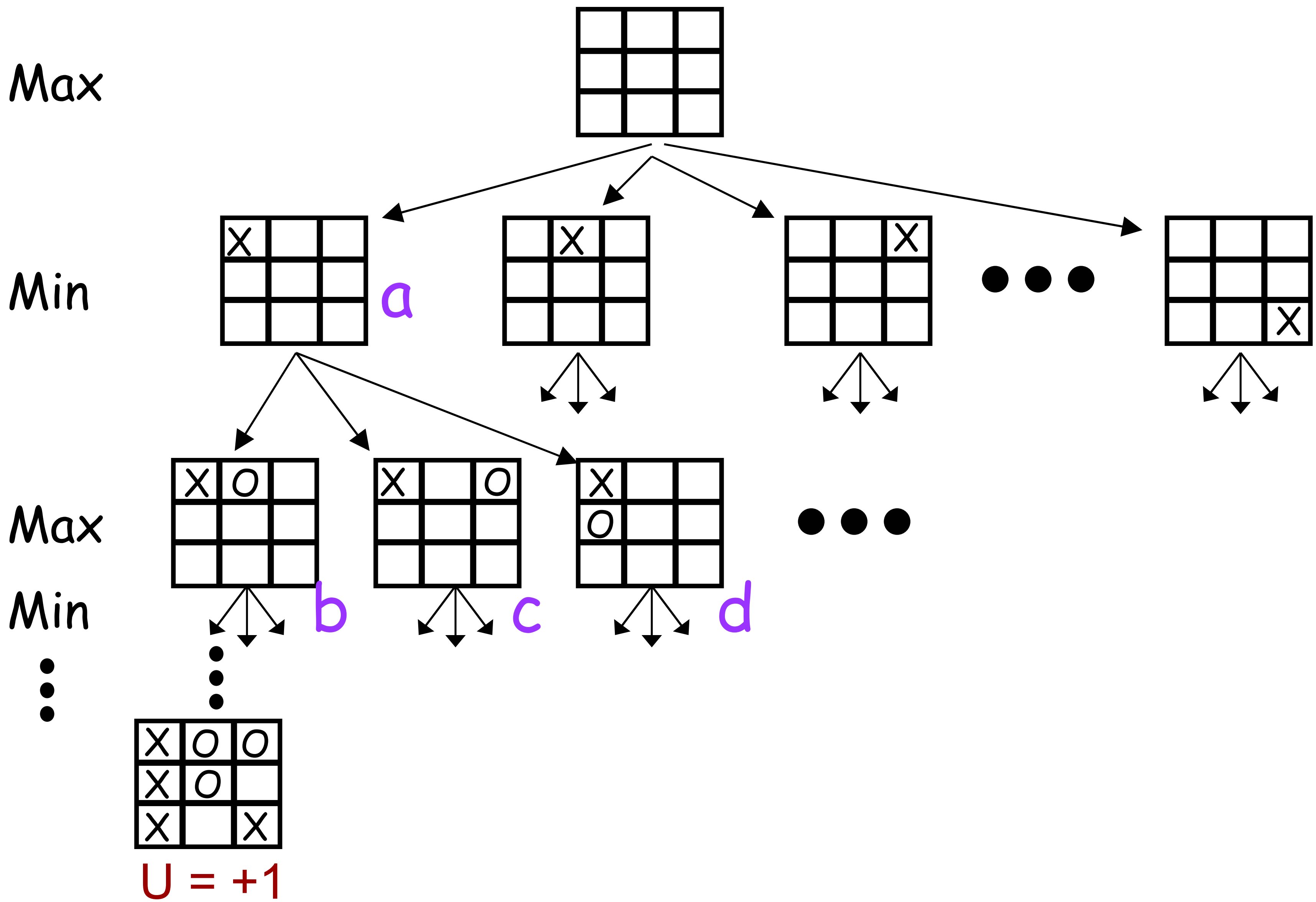
Intuitions

- Players alternate moves (starting with Max)
  - Game ends when some terminal p ϵ T is reached
- A game state: a position-player pair
  - tells us what position we're in, whose move it is
- Utility function and terminals replace goals
  - Max wants to maximize the terminal payoff
  - Min wants to minimize the terminal payoff
- Think of it as:
  - Max gets U(t), Min gets -U(t) for terminal node t
  - This is why it's called zero (or constant) sum

Tic-tac-toe: States



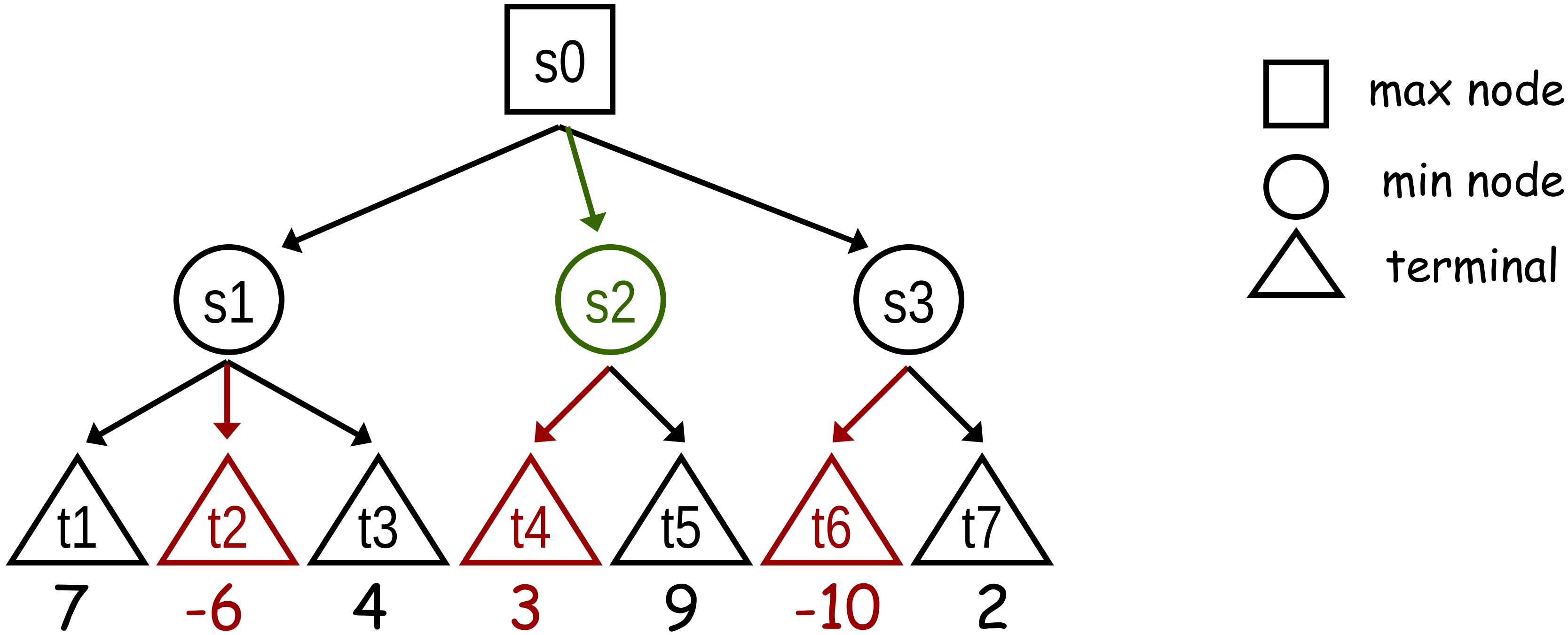
Tic-tac-toe: Game Tree



Game Tree

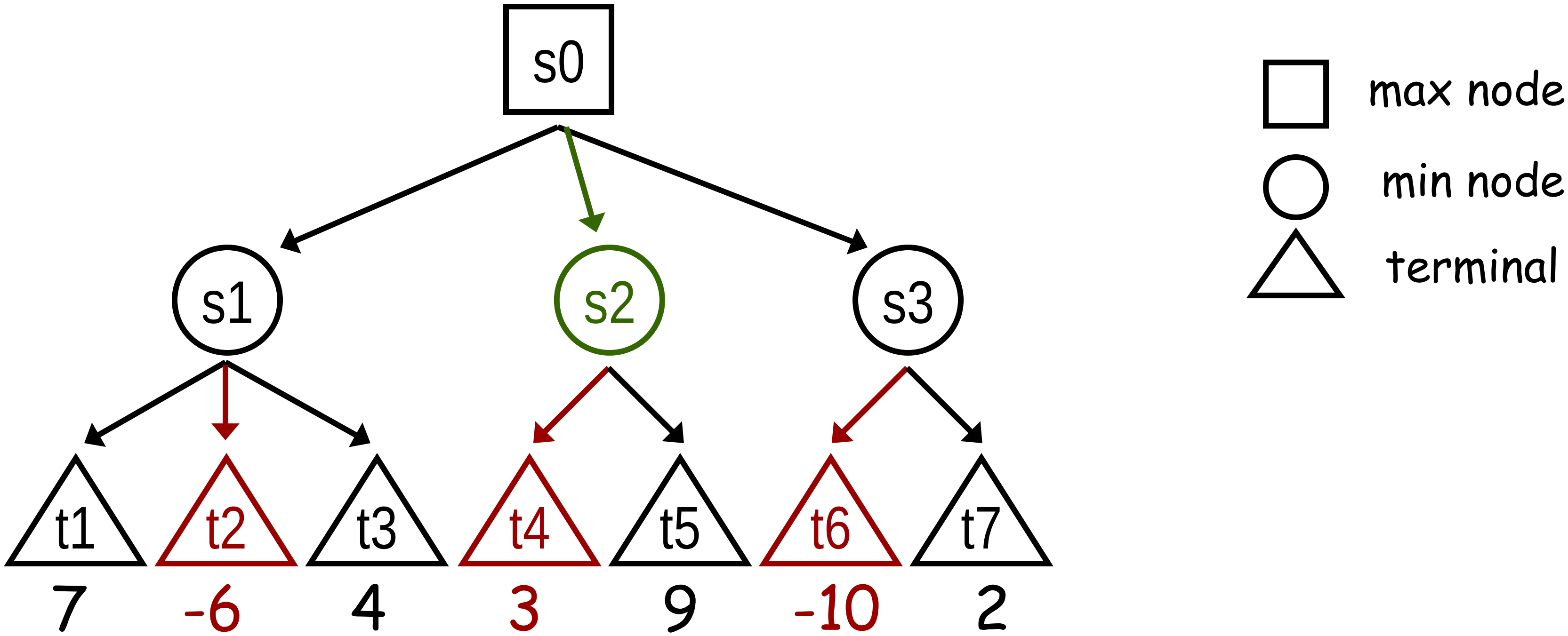
- Game tree looks like a search tree
  - Layers reflect the alternating moves
- But Max doesn't decide where to go alone
  - after Max moves to state a, Mins decides whether to move to state b, c, or d
- Thus Max must have a *strategy*
  - must know what to do next no matter what move Min makes (b, c, or d)
  - a sequence of moves will not suffice: Max may want to do something different in response to b, c, or d
- What is a *reasonable* strategy?

Minimax Strategy: Intuitions



The terminal nodes have utilities.
But we can compute a "utility" for the non-terminal states, by assuming both players always play their best move.

Minimax Strategy: Intuitions



If Max goes to s1, Min goes to t2
  * U(s1) = min{U(t1), U(t2), U(t3)} = -6
If Max goes to s2, Min goes to t4
  * U(s2) = min{U(t4), U(t5)} = 3
If Max goes to s3, Min goes to t6
  * U(s3) = min{U(t6), U(t7)} = -10

So Max goes to s2: so U(s0)
  = max{U(s1), U(s2), U(s3)}
  = 3

Minimax Strategy

- Build full game tree (all leaves are terminals)
  - root is start state, edges are possible moves, etc.
  - label terminal nodes with utilities
- Back values *up* the tree
  - $U(t)$ is defined for all terminals (part of input)
  - $U(n) =$ min $\{U(c) : c$ a child of $n\}$ if n is a min node
  - $U(n) =$ max $\{U(c) : c$ a child of $n\}$ if n is a max node

Minimax Strategy

- The values labeling each state are the values that Max will achieve in that state if both he and Min play their best moves.
  - Max plays a move to change the state to the highest valued min child.
  - Min plays a move to change the state to the lowest valued max child.
- If Min plays poorly, Max could do better, but never worse.
  - If Max, however know that Min will play poorly, there might be a better strategy of play for Max than minimax!