CS325 Artificial Intelligence
Ch. 5, Games!

Cengiz Günay, Emory Univ.

vs.


Spring 2013

AI in Games

A lot of work is done on it. Why?

AI in Games

A lot of work is done on it. Why?

- Fun, provide entertainment
- Also, simpler than life: toy problems

AI in Games

A lot of work is done on it. Why?

- Fun, provide entertainment
- Also, simpler than life: toy problems

Types of game AIs:

AI in Games

A lot of work is done on it. Why?

- Fun, provide entertainment
- Also, simpler than life: toy problems

Types of game AIs:

- Adversaries

zerg rush

AI in Games

A lot of work is done on it. Why?

- Fun, provide entertainment
- Also, simpler than life: toy problems

Types of game AIs:

- Adversaries
- Simulated reality (non-playable characters, world reaction to player).

zerg rush

AI in Games

A lot of work is done on it. Why?

- Fun, provide entertainment
- Also, simpler than life: toy problems

Types of game AIs:

- Adversaries
- Simulated reality (non-playable characters, world reaction to player).
- Game theory (next class)

zerg rush

Entry/Exit Surveys

Exit survey: Hidden Markov Models

- In the mining robot example, when is the uncertainty of the robot's trajectories reduced?
- How is Particle Filtering like and unlike a water filter?

Entry survey: Adversarial Games (0.25 points of final grade)

- What algorithm would be useful in games? Give examples with two different algorithms you learned in class.
- How would you help an agent solve a problem against an adversary? Think of a game like chess or checkers for starters.

Previously on AI for Games. . .

Can previous algorithms help in games?

- Single-state agent:

Previously on AI for Games. . .

Can previous algorithms help in games?

- Single-state agent: vacuum world, solving a maze

Previously on AI for Games. . .

Can previous algorithms help in games?

- Single-state agent: vacuum world, solving a maze
- Bayes Nets:

Previously on AI for Games...

Can previous algorithms help in games?

- Single-state agent: vacuum world, solving a maze
- Bayes Nets: card games

Previously on AI for Games...

Can previous algorithms help in games?

- Single-state agent: vacuum world, solving a maze
- Bayes Nets: card games
- Machine Learning:

Previously on AI for Games. . .

Can previous algorithms help in games?

- Single-state agent: vacuum world, solving a maze
- Bayes Nets: card games
- Machine Learning: guessing games, learning user moves

Previously on AI for Games...

Can previous algorithms help in games?

- Single-state agent: vacuum world, solving a maze
- Bayes Nets: card games
- Machine Learning: guessing games, learning user moves
- Logic, planning:

Previously on AI for Games...

Can previous algorithms help in games?

- Single-state agent: vacuum world, solving a maze
- Bayes Nets: card games
- Machine Learning: guessing games, learning user moves
- Logic, planning: board game with complex rules (Machinarium)

Previously on AI for Games. . .

Can previous algorithms help in games?

- Single-state agent: vacuum world, solving a maze
- Bayes Nets: card games
- Machine Learning: guessing games, learning user moves
- Logic, planning: board game with complex rules (Machinarium)
- MDPs, Reinforcement Learning:

Previously on AI for Games...

Can previous algorithms help in games?

- Single-state agent: vacuum world, solving a maze
- Bayes Nets: card games
- Machine Learning: guessing games, learning user moves
- Logic, planning: board game with complex rules (Machinarium)
- MDPs, Reinforcement Learning: pathfinding, optimal strategy for zerg

Previously on AI for Games. . .

Can previous algorithms help in games?

- Single-state agent: vacuum world, solving a maze
- Bayes Nets: card games
- Machine Learning: guessing games, learning user moves
- Logic, planning: board game with complex rules (Machinarium)
- MDPs, Reinforcement Learning: pathfinding, optimal strategy for zerg
- HMMs, Particle Filter:

Previously on AI for Games. . .

Can previous algorithms help in games?

- Single-state agent: vacuum world, solving a maze
- Bayes Nets: card games
- Machine Learning: guessing games, learning user moves
- Logic, planning: board game with complex rules (Machinarium)
- MDPs, Reinforcement Learning: pathfinding, optimal strategy for zerg
- HMMs, Particle Filter: state estimation and future prediction, partially observable environment with traps (sonic?)

Previously on AI for Games. . .

Can previous algorithms help in games?

- Single-state agent: vacuum world, solving a maze
- Bayes Nets: card games
- Machine Learning: guessing games, learning user moves
- Logic, planning: board game with complex rules (Machinarium)
- MDPs, Reinforcement Learning: pathfinding, optimal strategy for zerg
- HMMs, Particle Filter: state estimation and future prediction, partially observable environment with traps (sonic?)

None for adversaries?

Properties of Games as Agent Environment

Stochastic	Part.-Observ.	Unknown	Adversarial	Game
				Chess, Checkers
				Robot Soccer
				Poker
				Hide-and-go-seek
				Starcraft
				Battle for Wesnoth
				Halo/CoD/MoH
				Solitaire
				Minesweeper
				Zuma

Properties of Games as Agent Environment

Stochastic	Part.-Observ.	Unknown	Adversarial	Game
			X	Chess, Checkers
				Robot Soccer
				Poker
				Hide-and-go-seek
				Starcraft
				Battle for Wesnoth
				Halo/CoD/MoH
				Solitaire
				Minesweeper
				Zuma

Properties of Games as Agent Environment

Stochastic	Part.-Observ.	Unknown	Adversarial	Game
			X	Chess, Checkers
X	X		X	Robot Soccer
				Poker
				Hide-and-go-seek
				Starcraft
				Battle for Wesnoth
				Halo/CoD/MoH
				Solitaire
				Minesweeper
				Zuma

Properties of Games as Agent Environment

Stochastic	Part.-Observ.	Unknown	Adversarial	Game
			X	Chess, Checkers
X	X		X	Robot Soccer
	X		X	Poker
				Hide-and-go-seek
				Starcraft
				Battle for Wesnoth
				Halo/CoD/MoH
				Solitaire
				Minesweeper
				Zuma

Properties of Games as Agent Environment

Stochastic	Part.-Observ.	Unknown	Adversarial	Game
			X	Chess, Checkers
X	X		X	Robot Soccer
	X		X	Poker
X	X	X	X	Hide-and-go-seek
				Starcraft
				Battle for Wesnoth
				Halo/CoD/MoH
				Solitaire
				Minesweeper
				Zuma

Properties of Games as Agent Environment

Stochastic	Part.-Observ.	Unknown	Adversarial	Game
			X	Chess, Checkers
X	X		X	Robot Soccer
	X		X	Poker
X	X	X	X	Hide-and-go-seek
	X		X	Starcraft
				Battle for Wesnoth
				Halo/CoD/MoH
				Solitaire
				Minesweeper
				Zuma

Properties of Games as Agent Environment

Stochastic	Part.-Observ.	Unknown	Adversarial	Game
			X	Chess, Checkers
X	X		X	Robot Soccer
	X		X	Poker
X	X	X	X	Hide-and-go-seek
	X		X	Starcraft
X	X		X	Battle for Wesnoth
				Halo/CoD/MoH
				Solitaire
				Minesweeper
				Zuma

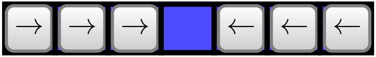
Properties of Games as Agent Environment

Stochastic	Part.-Observ.	Unknown	Adversarial	Game
			X	Chess, Checkers
X	X		X	Robot Soccer
	X		X	Poker
X	X	X	X	Hide-and-go-seek
	X		X	Starcraft
X	X		X	Battle for Wesnoth
X	X		X	Halo/CoD/MoH
				Solitaire
				Minesweeper
				Zuma

Properties of Games as Agent Environment

Stochastic	Part.-Observ.	Unknown	Adversarial	Game
			X	Chess, Checkers
X	X		X	Robot Soccer
	X		X	Poker
X	X	X	X	Hide-and-go-seek
	X		X	Starcraft
X	X		X	Battle for Wesnoth
X	X		X	Halo/CoD/MoH
	X			Solitaire
	X			Minesweeper
	X			Zuma

Single Player Games



Deterministic, single-state agent → Single-player game using **tree search**

- initial state
- player state
- possible actions
- results of actions
- utility values
- goal test

Adversarial Games

Adversarial Games

1. Start by adapting single-state agent to games
2. Define adversary as someone who wants you to lose
3. And makes decisions based on the outcome of your moves

Adversarial Games

Adversarial Games

1. Start by adapting single-state agent to games
2. Define adversary as someone who wants you to lose
3. And makes decisions based on the outcome of your moves

2-player games:

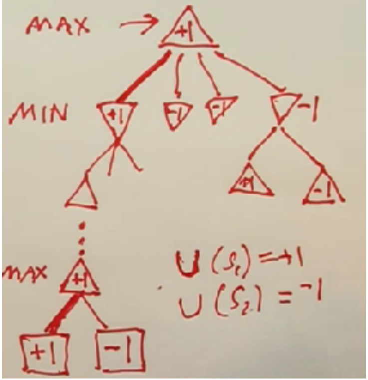
- Deterministic
- **Zero-sum**: Reward distributed between players
- **Minimax** algorithm: max & min players choose +/- utility, resp.

Adversarial Games

1. Start by adapting single-state agent to games
2. Define adversary as someone who wants you to lose
3. And makes decisions based on the outcome of your moves

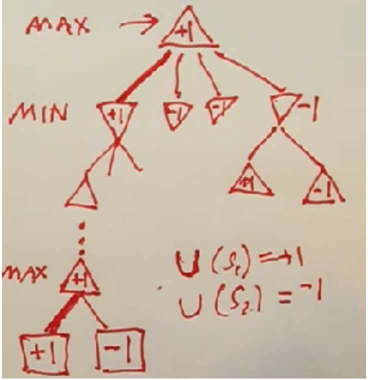
2-player games:

- Deterministic
- **Zero-sum**: Reward distributed between players
- **Minimax** algorithm: max & min players choose +/- utility, resp.



defun value(s):

- if s is $\Box$: $U(s)$
- if s is $\triangle$: $\mathrm{maxValue}(s)$
- if s is $\triangledown$: $\mathrm{minValue}(s)$



2-Player Value Function

defun $\mathrm{value}(s)$:

- if s is $\Box$: $U(s)$
- if s is $\triangle$: $\mathrm{maxValue}(s)$
- if s is ∇: $\mathrm{minValue}(s)$

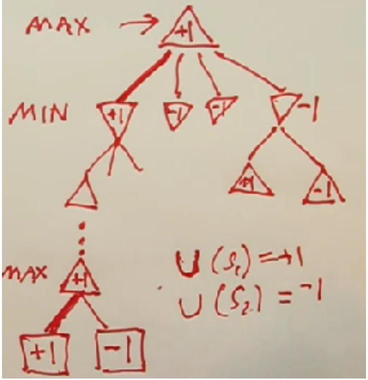
defun $m = \mathrm{maxValue}(s)$:

$m = -\infty$

for (a, s') in $\mathrm{successors}(s)$:

$v = \mathrm{value}(s')$

$m = \mathrm{maxValue}(m, v)$



2-Player Value Function

defun $\text{value}(s)$:

- if s is $\square$: $U(s)$
- if s is $\triangle$: $\text{maxValue}(s)$
- if s is $\triangledown$: $\text{minValue}(s)$

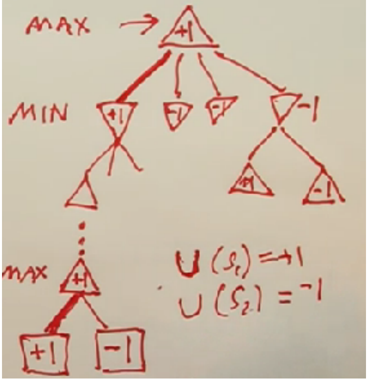
defun $m = \text{maxValue}(s)$:

$m = -\infty$

for (a, s') in $\text{successors}(s)$:

$v = \text{value}(s')$

$m = \text{maxValue}(m, v)$

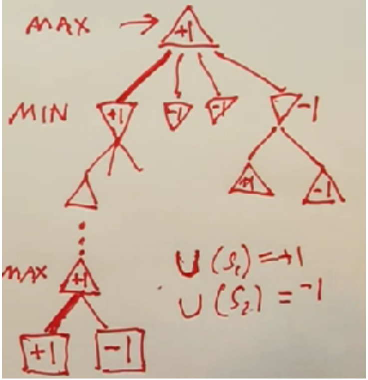


Assumes opponent is perfect!

Time Complexity

For a tree with
branching factor, b, and depth, m?

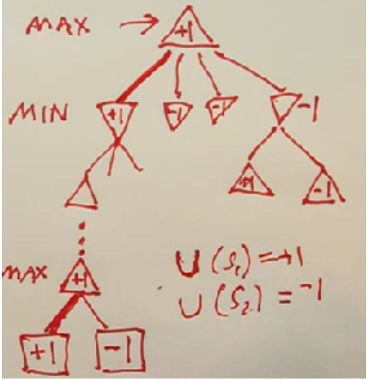
1. $O(bm)$
2. $O(b^m)$
3. $O(m^b)$



Time Complexity

For a tree with
branching factor, b, and depth, m?

1. $O(bm)$
2. $O(b^m)$
3. $O(m^b)$



Time Complexity

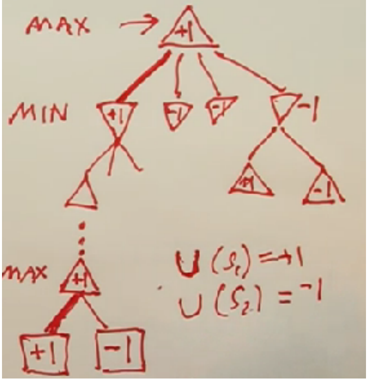
For a tree with
branching factor, b, and depth, m?

1. $O(bm)$
2. $O(b^m)$
3. $O(m^b)$

For chess: $b \simeq 30$, $m \simeq 40$
How long would it take with:

- 1 billion processors $\times 1$ billion/s evals?

1. seconds
2. minutes
3. hours
4. years
5. forever



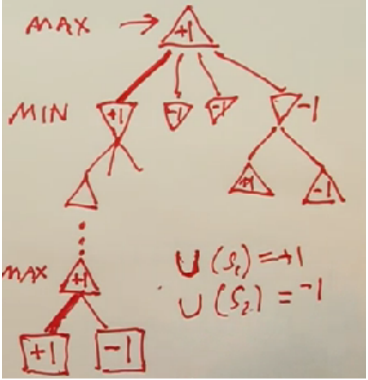
For a tree with
branching factor, b, and depth, m?

1. $O(bm)$
2. $O(b^m)$
3. $O(m^b)$

For chess: $b \simeq 30$, $m \simeq 40$
How long would it take with:

- 1 billion processors $\times 1$ billion/s evals?

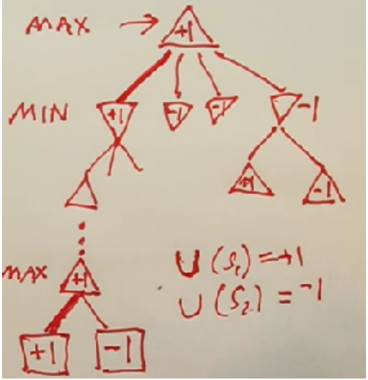
1. seconds
2. minutes
3. hours
4. years
5. **forever**



Space Complexity

For a tree with
branching factor, b, and depth, m?

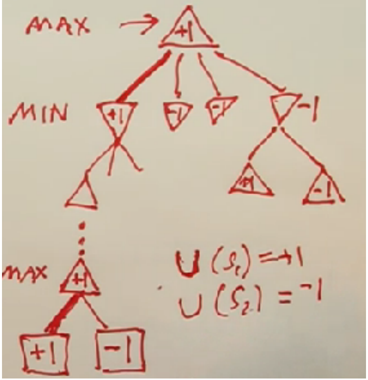
1. $O(bm)$
2. $O(b^m)$
3. $O(m^b)$



Space Complexity

For a tree with
branching factor, *b*, and depth, *m*?

1. $O(bm)$
2. $O(b^m)$
3. $O(m^b)$

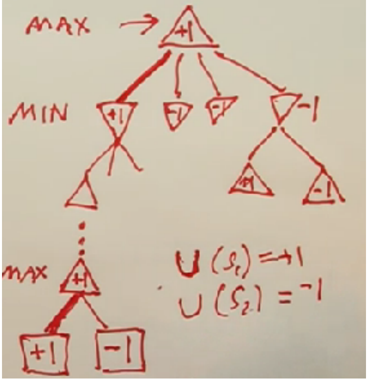


Space Complexity

For a tree with
branching factor, b, and depth, m?

1. $O(bm)$
2. $O(b^m)$
3. $O(m^b)$

Do not need more than total number of nodes.



Complexity Reduction?

How to do it?

1. Reduce b
2. Reduce m
3. Tree $\rightarrow$ graph

Complexity Reduction?

How to do it?

1. Reduce b
2. Reduce m
3. Tree $\rightarrow$ graph

All of the above!

Example

defun value(s):

- if s is $\Box$: $U(s)$
- if s is $\triangle$: $\text{maxValue}(s)$
- if s is ∇: $\text{minValue}(s)$

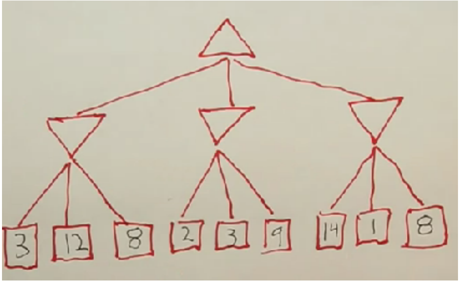
defun $m = \text{maxValue}(s)$:

$m = -\infty$

for (a, s') in $\text{successors}(s)$:

$v = \text{value}(s')$

$m = \text{maxValue}(m, v)$

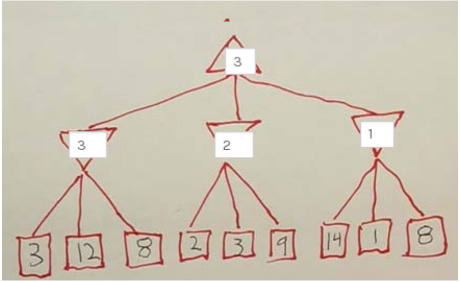


Example

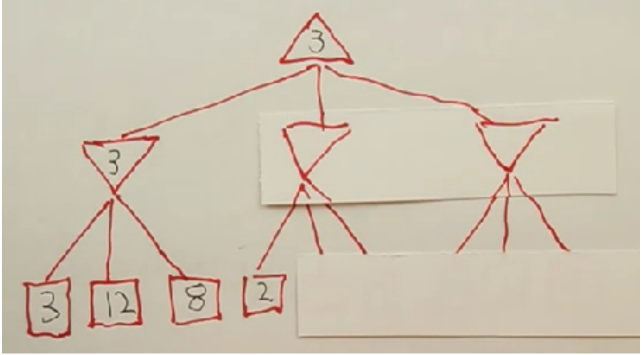
defun value(s):

- if s is $\square$: $U(s)$
- if s is $\triangle$: $\mathrm{maxValue}(s)$
- if s is $\triangledown$: $\mathrm{minValue}(s)$

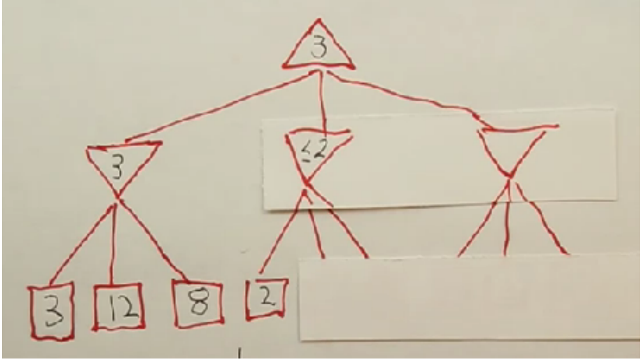
defun $m = \mathrm{maxValue}(s)$:
  $m = -\infty$
  for (a, s') in $\mathrm{successors}(s)$:
    $v = \mathrm{value}(s')$
    $m = \mathrm{maxValue}(m, v)$



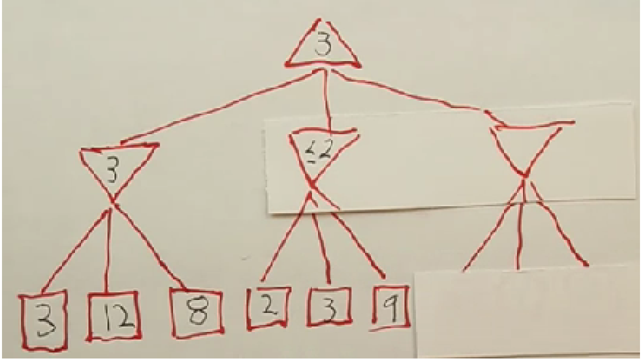
Reducing Branching Factor, b



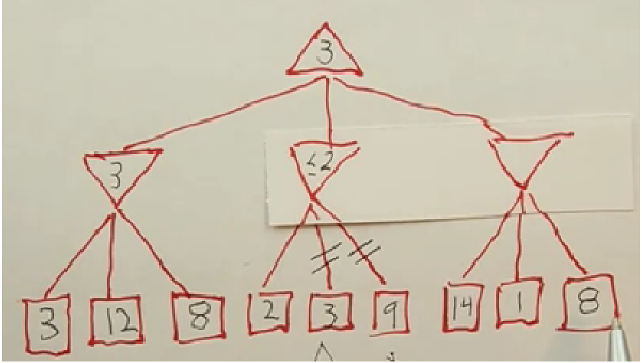
Reducing Branching Factor, b



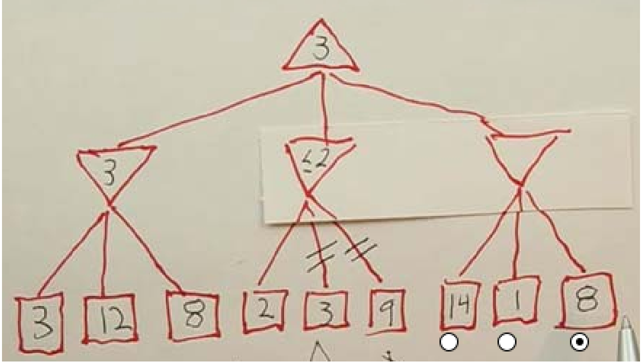
Reducing Branching Factor, b



Reducing Branching Factor, b



Reducing Branching Factor, *b*



Which one to prune?

Reducing Depth, m

Select a cutoff:

- Limit m (e.g., plan 3 steps ahead in chess)
- Estimate terminal nodes' utility with evaluation function
  - like heuristics
- Learn from experience
- In chess, use board state, value of pieces, etc.
- For value of pieces: $\text{eval}(s) = \sum_i w_i p_i$
  - can use machine learning for w_i

Formalize as Alpha-Beta Pruning

defun $\mathrm{value}(s)$:
  cutoff at depth m': $\mathrm{eval}(s)$
  if s is $\square$: $U(s)$
  if s is $\triangle$: $\mathrm{maxValue}(s, depth, \alpha, \beta)$
  if s is ∇: $\mathrm{minValue}(s, depth, \alpha, \beta)$
where α, β are overall max and min values, resp.

Formalize as Alpha-Beta Pruning

defun $\text{value}(s)$:
  cutoff at depth m': $\text{eval}(s)$
  if s is $\square$: $U(s)$
  if s is $\triangle$: $\text{maxValue}(s, \mathit{depth}, \alpha, \beta)$
  if s is ∇: $\text{minValue}(s, \mathit{depth}, \alpha, \beta)$
where α, β are overall max and min values, resp.

defun $v = \text{maxValue}(s, \mathit{depth}, \alpha, \beta)$:
  $v = -\infty$
  for (a, s') in $\text{successors}(s)$:
    $v = \max(v, \text{value}(s', \mathit{depth} + 1, \alpha, \beta))$
    if $v > \beta$ return v
    $\alpha = \max(\alpha, v)$

Formalize as Alpha-Beta Pruning

defun $\text{value}(s)$:
  cutoff at depth m': $\text{eval}(s)$
  if s is $\square$: $U(s)$
  if s is $\triangle$: $\text{maxValue}(s, \textit{depth}, \alpha, \beta)$
  if s is ∇: $\text{minValue}(s, \textit{depth}, \alpha, \beta)$
where α, β are overall max and min values, resp.

defun $v = \text{maxValue}(s, \textit{depth}, \alpha, \beta)$:
  $v = -\infty$
  for (a, s') in $\text{successors}(s)$:
    $v = \max(v, \text{value}(s', \textit{depth} + 1, \alpha, \beta))$
    if $v > \beta$ **return** v
    $\alpha = \max(\alpha, v)$

Formalize as Alpha-Beta Pruning

defun $\mathrm{value}(s)$:
  cutoff at depth m': $\mathrm{eval}(s)$
  if s is □: $U(s)$
  if s is △: $\mathrm{maxValue}(s, depth, \alpha, \beta)$
  if s is ▽: $\mathrm{minValue}(s, depth, \alpha, \beta)$
where α, β are overall max and min values, resp.

defun $v = \mathrm{maxValue}(s, depth, \alpha, \beta)$:
  $v = -\infty$
  for (a, s') in $\mathrm{successors}(s)$:
    $v = \max(v, \mathrm{value}(s', depth + 1, \alpha, \beta))$
    if $v > \beta$ return v
    $\alpha = \max(\alpha, v)$

Can cut up to $O(b^{m/2})$!

Complexity Reduction by Tree → Graph

Convert into graph search problem:

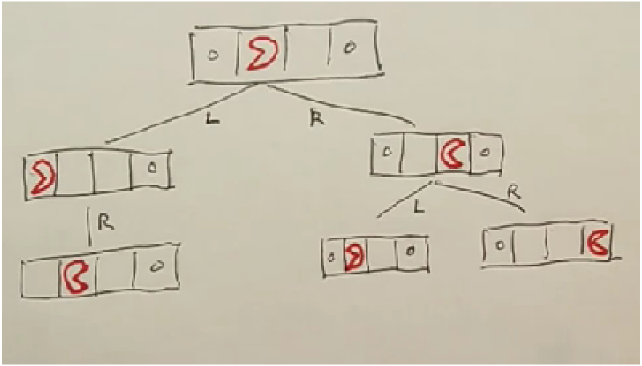
- to reach special **opening and closing states**
- to make and protect from **killer-moves**

Complexity Reduction by Tree → Graph

Convert into graph search problem:

- to reach special **opening and closing states**
- to make and protect from **killer-moves**

Utility: How many food particles can pacman eat?

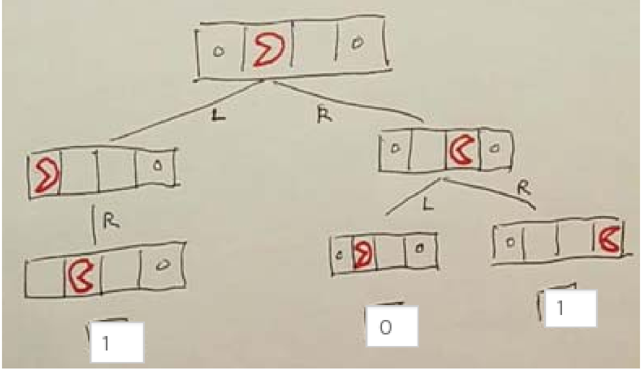


Complexity Reduction by Tree → Graph

Convert into graph search problem:

- to reach special **opening and closing states**
- to make and protect from **killer-moves**

Utility: How many food particles can pacman eat?

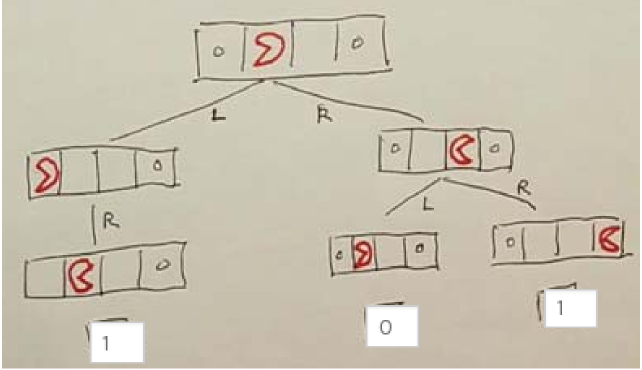


Complexity Reduction by Tree → Graph

Convert into graph search problem:

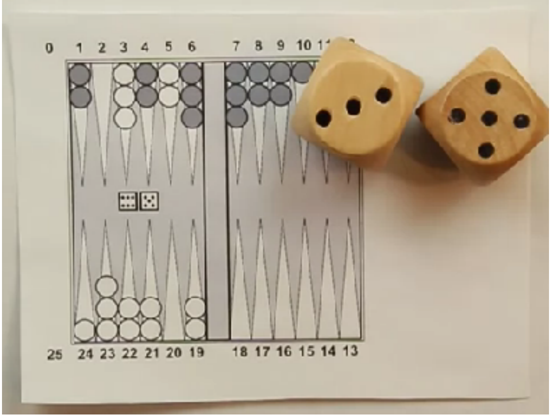
- to reach special **opening and closing states**
- to make and protect from **killer-moves**

Utility: How many food particles can pacman eat?



2-step limit causes **horizon effect**?

Stochastic Games



Stochastic Games

defun value(s):

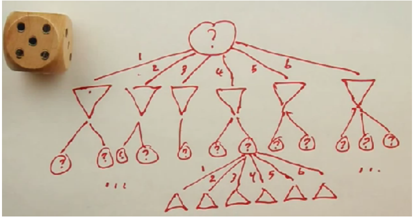
cutoff at depth m': $\text{eval}(s)$

if s is $\square$: $U(s)$

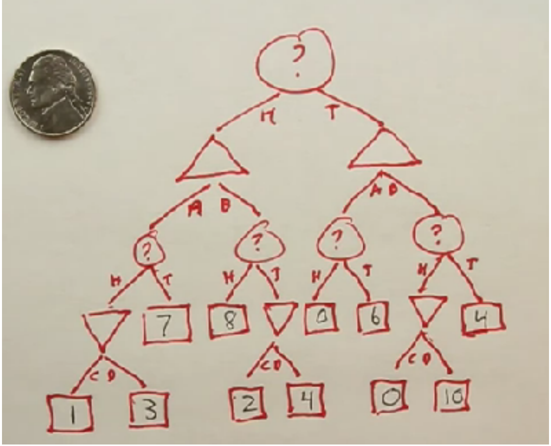
if s is $\triangle$: $\text{maxValue}(s, \textit{depth}, \alpha, \beta)$

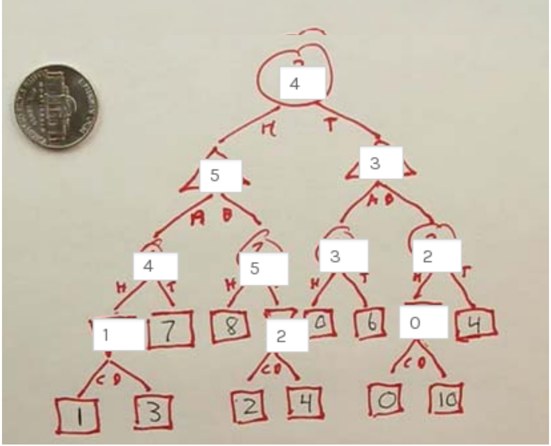
if s is $\triangledown$: $\text{minValue}(s, \textit{depth}, \alpha, \beta)$

if s is ?: $\text{expValue}(s, \textit{depth}, \alpha, \beta)$



Coin-flip Game




4
H
T
5
3
A
B
4
5
3
2
1
7
8
2
0
6
0
4
1
3
2
4
0
10