UIT2504 Artificial Intelligence

Search and Non-determinism

C. Aravindan
<AravindanC@ssn.edu.in>

Professor of Information Technology
SSN College of Engineering

September 19, 2024

Complex Environments

- We have so far assumed a fully observable, deterministic, and known environments

Complex Environments

- We have so far assumed a fully observable, deterministic, and known environments
- We need to understand techniques for problem solving (or game playing) in more challenging environments, such as those that are only partially observable or nondeterministic

Complex Environments

- We have so far assumed a fully observable, deterministic, and known environments
- We need to understand techniques for problem solving (or game playing) in more challenging environments, such as those that are only partially observable or nondeterministic
- We will first deal with the nondeterminism
- When an agent performs an action on the environment, there is no deterministic resulting state, but a set of possible resulting states

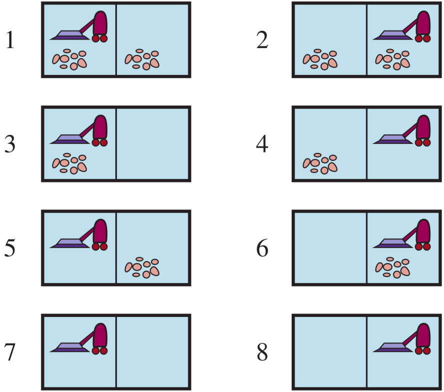
Complex Environments

- We have so far assumed a fully observable, deterministic, and known environments
- We need to understand techniques for problem solving (or game playing) in more challenging environments, such as those that are only partially observable or nondeterministic
- We will first deal with the nondeterminism
- When an agent performs an action on the environment, there is no deterministic resulting state, but a set of possible resulting states
- A set of states that the agent believes are possible is called as a belief state

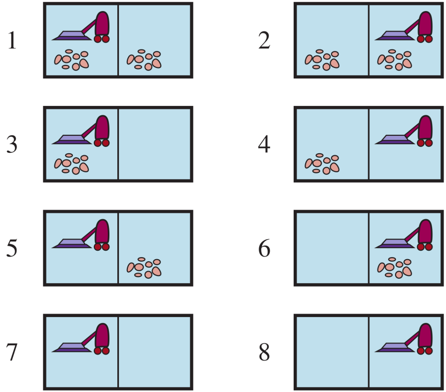
Complex Environments

- We have so far assumed a fully observable, deterministic, and known environments
- We need to understand techniques for problem solving (or game playing) in more challenging environments, such as those that are only partially observable or nondeterministic
- We will first deal with the nondeterminism
- When an agent performs an action on the environment, there is no deterministic resulting state, but a set of possible resulting states
- A set of states that the agent believes are possible is called as a belief state
- So, the agent is in one belief state and by performing an action moves to another belief state

Vacuum World Revisited

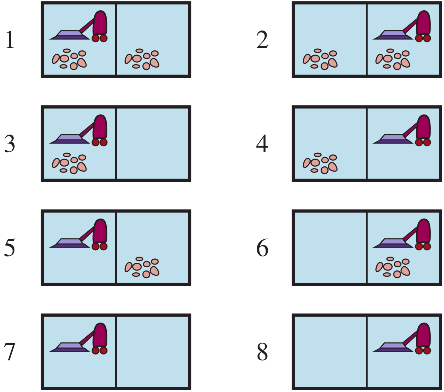


Vacuum World Revisited



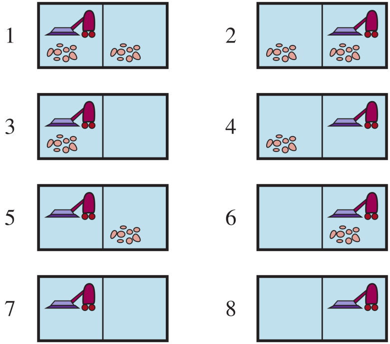
- Consider three possible actions — *Right*, *Left*, and *Suck*

Vacuum World Revisited

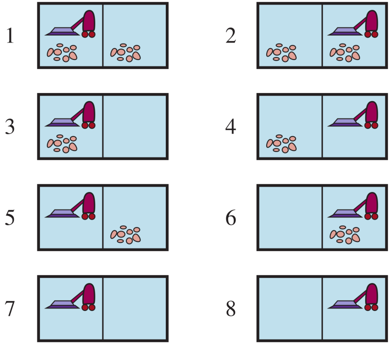


- Consider three possible actions — *Right*, *Left*, and *Suck*
- If the environment is simple, and the initial state is 1, then any search technique can be used to find a goal state (7 or 8) — [*Suck*, *Right*, *Suck*]

Erratic Vacuum World

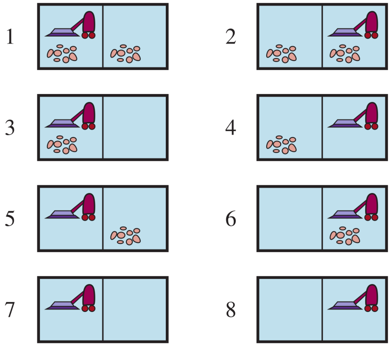


Erratic Vacuum World

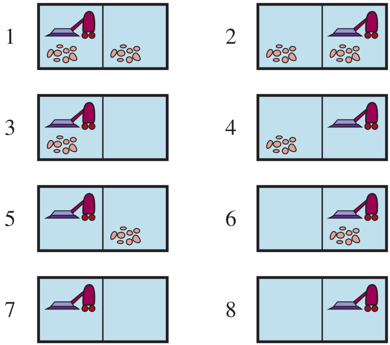


- Suppose, we introduce nondeterminism through erratic behavior of the *Suck* action:
  - When applied to a dirty square the action cleans the square and sometimes cleans up dirt in the adjacent square too
  - When applied to a clean square the action sometimes deposits dirt on the carpet

Erratic Vacuum World

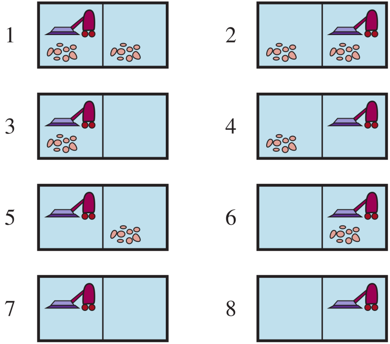


Erratic Vacuum World



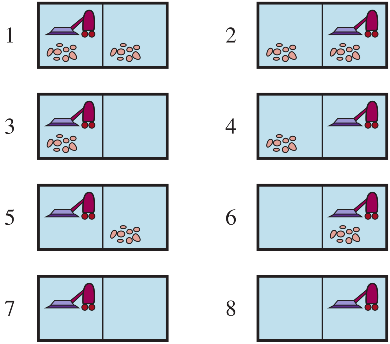
- This can be captured by appropriately modifying the transition model

Erratic Vacuum World

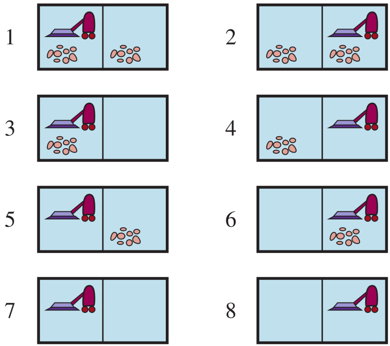


- This can be captured by appropriately modifying the transition model
- $RESULTS(1, Suck) = \{5, 7\}$

Erratic Vacuum World

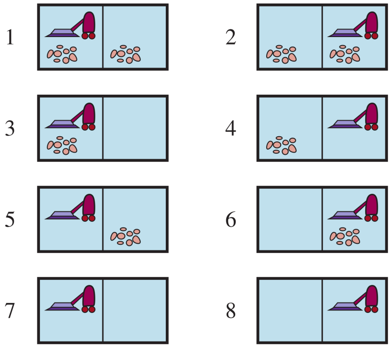


Erratic Vacuum World



- Suppose, we start from state 1, what may be a solution? Is it a sequence of action?

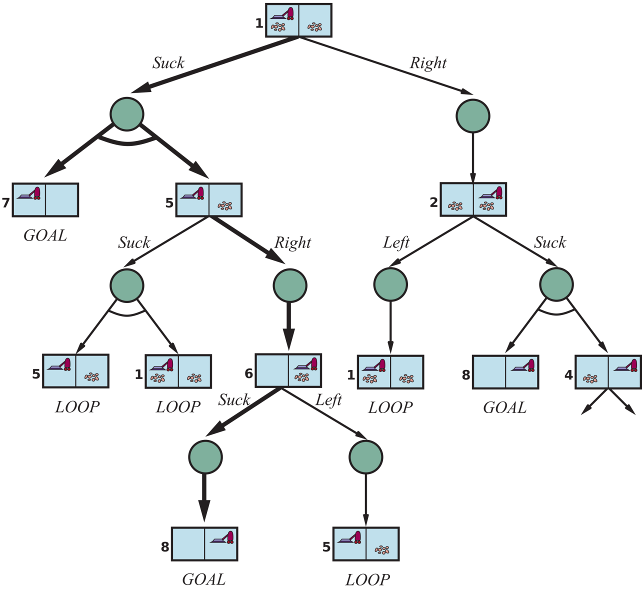
Erratic Vacuum World



- Suppose, we start from state 1, what may be a solution? Is it a sequence of action?
- The solution now is a tree! It is called as a condition plan (also known as a contingency plan or a strategy)

$$[\mathit{Suck}, \textbf{if } \mathit{State} = 5 \textbf{ then } [\mathit{Right}, \mathit{Suck}] \textbf{ else } []]$$

AND–OR Search Trees



- As we can see, solution is no more a path in the search tree

Solution in a AND–OR Tree

- As we can see, solution is no more a path in the search tree
- Solution is a subtree that
  - has a goal node at each leaf
  - specifies one action at each of its OR nodes
  - includes every outcome branch at each of its AND nodes

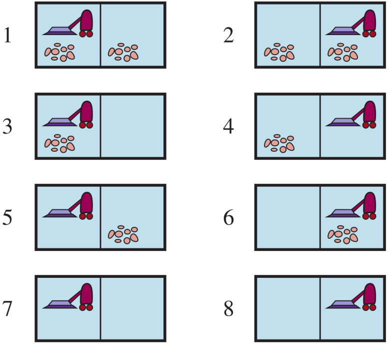
AND–OR Search Algorithm

function AND-OR-SEARCH(*problem*) **returns** a conditional plan, or *failure*
  return OR-SEARCH(*problem*, *problem*.INITIAL, [ ])

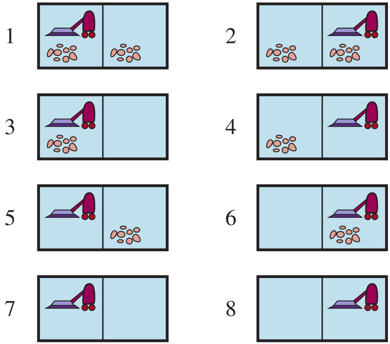
function OR-SEARCH(*problem*, *state*, *path*) **returns** *a conditional plan, or failure*
  if *problem*.IS-GOAL(*state*) **then return** the empty plan
  if IS-CYCLE(*path*) **then return** *failure*
  for each *action* **in** *problem*.ACTIONS(*state*) **do**
    plan ← AND-SEARCH(*problem*, RESULTS(*state*, *action*), [*state*] + *path*])
    if *plan* ≠ *failure* **then return** [*action*] + *plan*]
  return *failure*

function AND-SEARCH(*problem*, *states*, *path*) **returns** *a conditional plan, or failure*
  for each s_i **in** *states* **do**
    $plan_i$ ← OR-SEARCH(*problem*, s_i, *path*)
    if $plan_i$ = *failure* **then return** *failure*
  return [**if** s_1 **then** $plan_1$ **else if** s_2 **then** $plan_2$ **else** $\ldots$ **if** s_{n-1} **then** $plan_{n-1}$ **else** $plan_n$]

Slippery Vacuum World

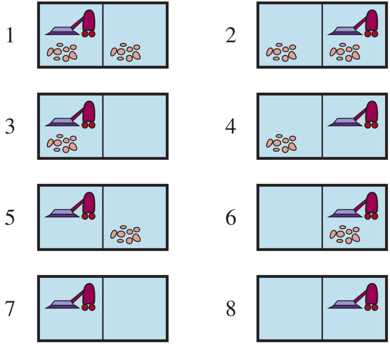


Slippery Vacuum World



- Consider a different variation of the conventional vacuum world problem — movement actions sometimes fail, leaving the agent in the same location
- This can be captured by appropriately modifying the transition model

Slippery Vacuum World



- Consider a different variation of the conventional vacuum world problem — movement actions sometimes fail, leaving the agent in the same location
- This can be captured by appropriately modifying the transition model
- $RESULTS(1, Right) = \{1, 2\}$, $RESULTS(2, Left) = \{1, 2\}$, and so on

Cyclic Plans

- Suppose we start from state 1, the AND–OR algorithm fails to find a conditional plan

Cyclic Plans

- Suppose we start from state 1, the AND–OR algorithm fails to find a conditional plan
- However, there is a cyclic solution that an agent may adopt — keep trying *Right* until it works

$$[Suck, \ \textbf{while } State = 5 \textbf{ do } Right, Suck]$$

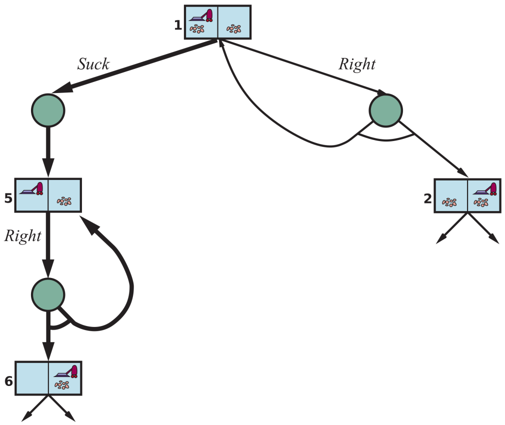
Cyclic Plans

- Suppose we start from state 1, the AND–OR algorithm fails to find a conditional plan
- However, there is a cyclic solution that an agent may adopt — keep trying *Right* until it works

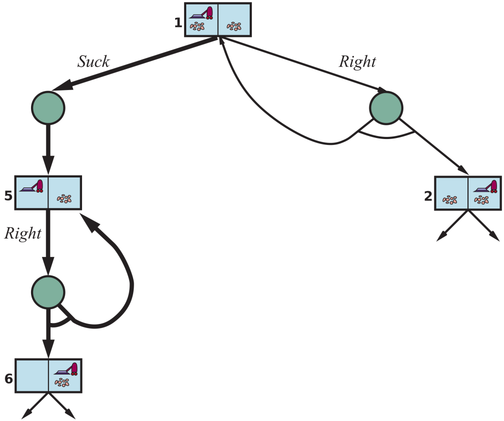
$$[\mathit{Suck},\ \textbf{while}\ \mathit{State} = 5\ \textbf{do}\ \mathit{Right}, \mathit{Suck}]$$

- This idea works if the driving mechanism is just slippery and works otherwise. The plan does not work if the agent can never move right

Cyclic Plans



Cyclic Plans



- Solution is a “tree” with possible loops, where every leaf is a goal state and that a leaf is reachable from every point in the plan

Questions?

Stochastic Games

- There are games such as Backgammon that are nondeterministic in nature — roll of dice determines possible set of actions in a given state

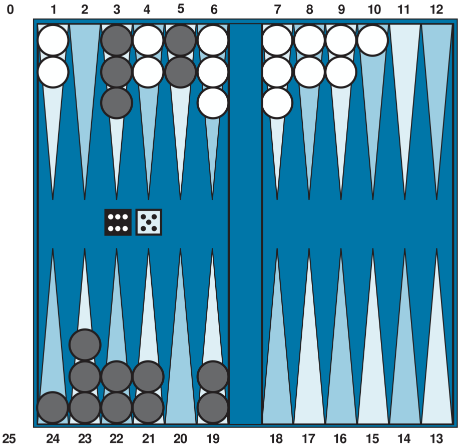
Stochastic Games

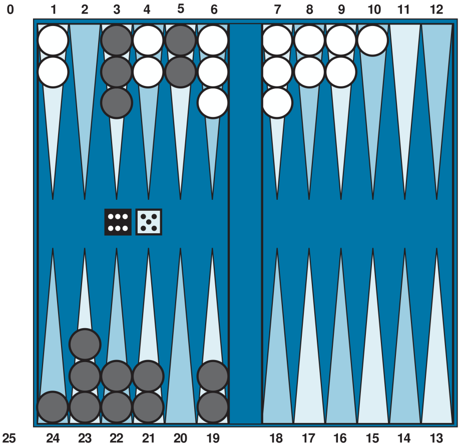
- There are games such as Backgammon that are nondeterministic in nature — roll of dice determines possible set of actions in a given state
- Depending on what has been rolled, a player, say Black, has certain choices of actions but can not think ahead as nobody knows what will be the result of White's dice roll

Stochastic Games

- There are games such as Backgammon that are nondeterministic in nature — roll of dice determines possible set of actions in a given state
- Depending on what has been rolled, a player, say Black, has certain choices of actions but can not think ahead as nobody knows what will be the result of White's dice roll
- The game tree is similar to the AND-OR tree, but basic probability knowledge can be used

Stochastic Games





- Black, which moves clockwise from 0 to 25, has four possible moves for this roll of dice (6-5): (5–11, 5–10), (5–11, 19–24), (5–10, 10–16), (5–11, 11–16)

Game Tree with Chance Nodes

- After Black makes a move, there is nondeterminism due to dice roll, and that is captured by a chance node

Game Tree with Chance Nodes

- After Black makes a move, there is nondeterminism due to dice roll, and that is captured by a chance node
- How many possible outcomes are there for a chance node?

Game Tree with Chance Nodes

- After Black makes a move, there is nondeterminism due to dice roll, and that is captured by a chance node
- How many possible outcomes are there for a chance node? 21 distinct rolls out of 36 possible outcomes

Game Tree with Chance Nodes

- After Black makes a move, there is nondeterminism due to dice roll, and that is captured by a chance node
- How many possible outcomes are there for a chance node? 21 distinct rolls out of 36 possible outcomes
- What is the probability of each roll?

Game Tree with Chance Nodes

- After Black makes a move, there is nondeterminism due to dice roll, and that is captured by a chance node
- How many possible outcomes are there for a chance node? 21 distinct rolls out of 36 possible outcomes
- What is the probability of each roll? All rolls where both the dice have the same value have probability $\frac{1}{36}$, whereas all the other 15 distinct rolls have probability $\frac{1}{18}$

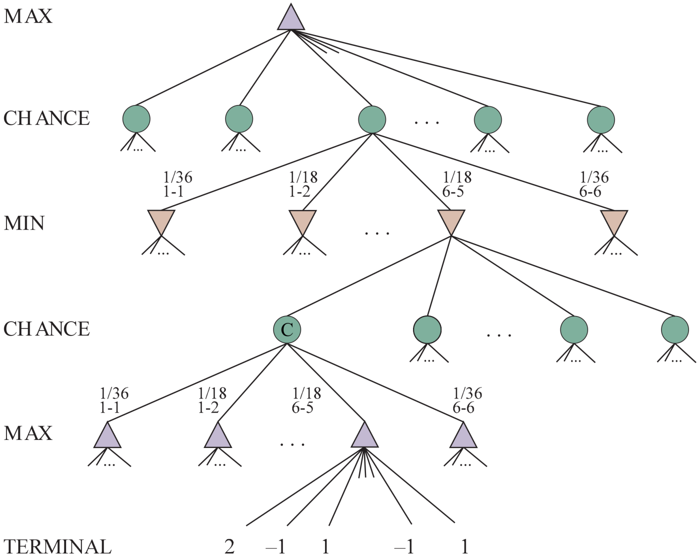
Game Tree with Chance Nodes

- After Black makes a move, there is nondeterminism due to dice roll, and that is captured by a chance node
- How many possible outcomes are there for a chance node? 21 distinct rolls out of 36 possible outcomes
- What is the probability of each roll? All rolls where both the dice have the same value have probability $\frac{1}{36}$, whereas all the other 15 distinct rolls have probability $\frac{1}{18}$
- So every chance node has a branching factor of 21 and edges are labelled with probability values instead of actions

Game Tree with Chance Nodes

- After Black makes a move, there is nondeterminism due to dice roll, and that is captured by a chance node
- How many possible outcomes are there for a chance node? 21 distinct rolls out of 36 possible outcomes
- What is the probability of each roll? All rolls where both the dice have the same value have probability $\frac{1}{36}$, whereas all the other 15 distinct rolls have probability $\frac{1}{18}$
- So every chance node has a branching factor of 21 and edges are labelled with probability values instead of actions
- We make a weighted sum of utility values of all the children to get the expected utility of a chance node

Game Tree with Chance Nodes



Expectiminimax

- Now, the idea of minimax value for deterministic games can be generalized to the expectiminimax value as follows:

  EXPECTIMINIMAX(s) =

$$\begin{cases} Utility(s, MAX) & \text{if IS\_TERMINAL}(s) \\ max_a EMINIMAX(RESULT(s, a)) & \text{if TO\_MOVE}(s) = \text{MAX} \\ min_a EMINIMAX(RESULT(s, a)) & \text{if TO\_MOVE}(s) = \text{MIN} \\ \Sigma_r P(r) EMINIMAX(RESULT(s, r)) & \text{if TO\_MOVE}(s) = \text{CHANCE} \end{cases}$$

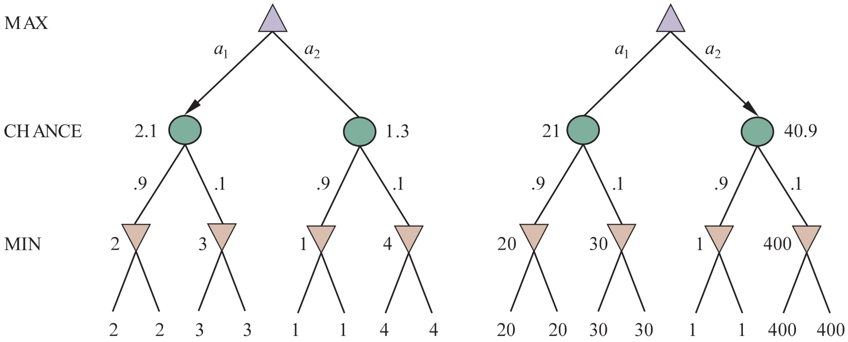
Cutoff and Evaluation Functions

- Like in the case of normal minimax algorithm, it is possible to introduce cutoff and heuristic evaluation function (to estimate the utility value of any arbitrary state)

Cutoff and Evaluation Functions

- Like in the case of normal minimax algorithm, it is possible to introduce cutoff and heuristic evaluation function (to estimate the utility value of any arbitrary state)
- However, the estimated values should be in agreement with the probability of winning (or, of the expected utility)

Cutoff and Evaluation Functions



Pruning

- What is the complexity of expectiminimax?

Pruning

- What is the complexity of expectiminimax? $O(b^m n^m)$

Pruning

- What is the complexity of expectiminimax? $O(b^m n^m)$
- Even if the search is restricted to some small depth d, the extra cost over minimax is significant

Pruning

- What is the complexity of expectiminimax? $O(b^m n^m)$
- Even if the search is restricted to some small depth d, the extra cost over minimax is significant
- It is possible to use the idea of alpha-beta pruning to the MIN and MAX nodes

Pruning

- What is the complexity of expectiminimax? $O(b^m n^m)$
- Even if the search is restricted to some small depth d, the extra cost over minimax is significant
- It is possible to use the idea of alpha-beta pruning to the MIN and MAX nodes
- Is it possible to prune a CHANCE node?

Pruning

- What is the complexity of expectiminimax? $O(b^m n^m)$
- Even if the search is restricted to some small depth d, the extra cost over minimax is significant
- It is possible to use the idea of alpha-beta pruning to the MIN and MAX nodes
- Is it possible to prune a CHANCE node? — bounds on the utility values can be extended to find bounds on the average

Pruning

- What is the complexity of expectiminimax? $O(b^m n^m)$
- Even if the search is restricted to some small depth d, the extra cost over minimax is significant
- It is possible to use the idea of alpha-beta pruning to the MIN and MAX nodes
- Is it possible to prune a CHANCE node? — bounds on the utility values can be extended to find bounds on the average
- Consider a game where 5 dices are rolled! The branching factor is too high. A Type B strategy of forward pruning may be applied

Pruning

- What is the complexity of expectiminimax? $O(b^m n^m)$
- Even if the search is restricted to some small depth d, the extra cost over minimax is significant
- It is possible to use the idea of alpha-beta pruning to the MIN and MAX nodes
- Is it possible to prune a CHANCE node? — bounds on the utility values can be extended to find bounds on the average
- Consider a game where 5 dices are rolled! The branching factor is too high. A Type B strategy of forward pruning may be applied
- Monte Carlo Tree Search is definitely possible as we can always take a random outcome for a dice roll!

Questions?