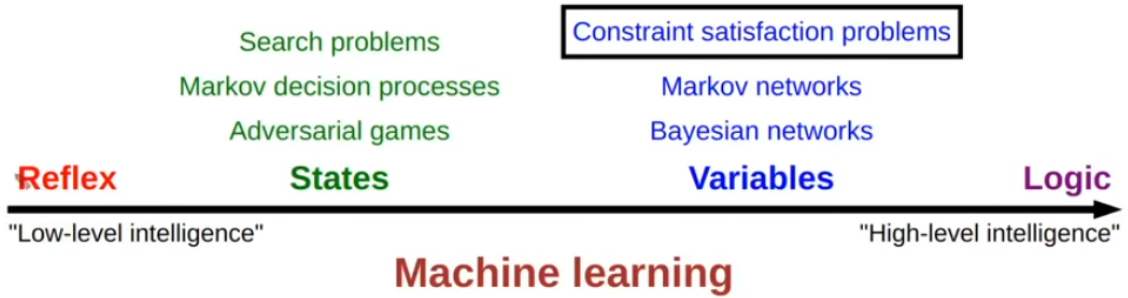
IE0005 Review

Content

1. Agent and Environment
2. Search
3. Constraint Satisfaction & Game Playing
4. Reinforcement Learning

Legend:

- **Core idea (for quiz)**
- **Extra insights**



Stanford AI: https://www.youtube.com/playlist?list=PLoROMvodv4rOca_Ovz1DvdtWuz8BfSWL2

Agent & Environment

Accessible (Full knowledge)	Knowledge of the environment	Inaccessible (Partial knowledge)
Deterministic (No)	Consequence of action or state change has a probability distribution?	Non-deterministic (Yes, have distribution)
Episodic	Previous action/perception affect the next?	Sequential
Static	Changing environment	Dynamic
Discrete	State and Action data type	Continuous

Non-deterministic example
Critical hit

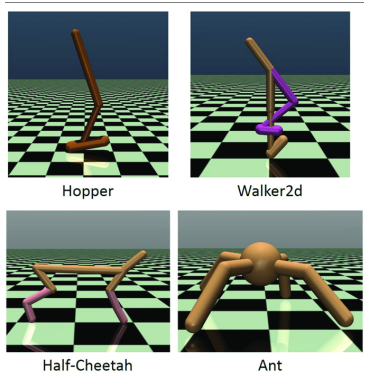
Dynamic example
Agent deliberation at 10 Hz
Environment changes at 20 Hz

Agent & Environment

Accessible	Inaccessible
Deterministic	Non-deterministic
Episodic	Sequential
Static	Dynamic
Discrete	Continuous

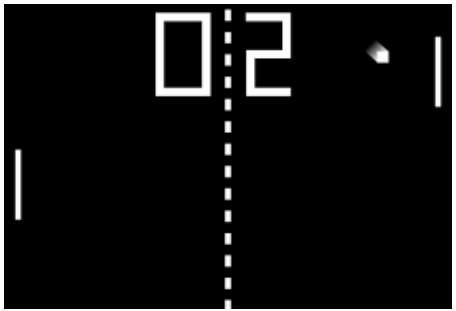
Chess

Space Invader

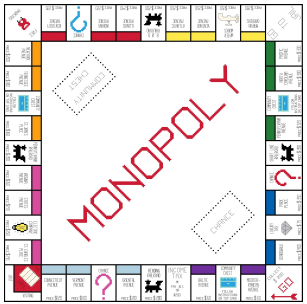


Walking bot simulation

Slot Machine



Pong



Monopoly

Starcraft

Questions

Accessible	Inaccessible
Deterministic	Non-deterministic
Episodic	Sequential
Static	Dynamic
Discrete	Continuous

Go

1. Driverless taxi is an accessible environment.
   a) True b)False

2. A slot machine playing agent operates in a sequential environment.
   a) True b)False

3. A Go playing agent operates in a dynamic task environment.
   a)True b)False

4. Chess is a deterministic environment.
   a)True b)False

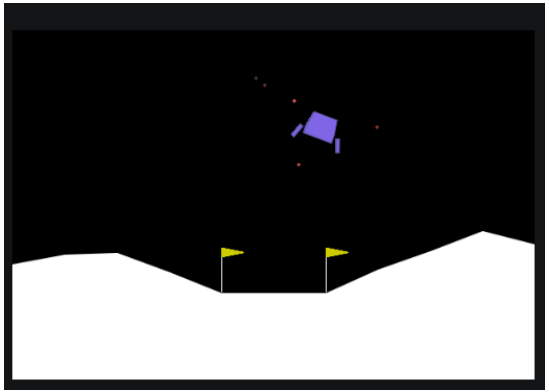
5. Minesweeper is a discrete environment.
   a)True b)False

6. Go is a discrete environment.
   a)True b)False

7. An agent that senses only partial information about the state can be rational.
   a)True b)False

Agent & Environment

- Why the “Agent and Environment” formulation?
  - Separate the “Agent logic” and “Environment logic”
  - Environment logic can be simple (If/else tree) or complex (simulations with physics)
  - Different agent logic can operate on the same environment. Agent logic developed for 1 environment might not be transferable



OpenAI Gym: https://www.gymlibrary.dev/index.html

The Gym interface is simple, pythonic, and capable of representing general RL problems:

```
import gym
env = gym.make("LunarLander-v2", render_mode="human")
observation, info = env.reset(seed=42)
for _ in range(1000):
   action = policy(observation)  # User-defined policy function
   observation, reward, terminated, truncated, info = env.step(action)

   if terminated or truncated:
      observation, info = env.reset()
env.close()
```

Environment logic

Agent logic / Policy

Agent & Environment

- Why the different classifications?
  - Different classifications have different information to be utilized
    - Full knowledge: find the optimal outcome
    - Partial knowledge: find the best guess at the moment
  - Allow us to program the agent logic, assign variables, correctly
    - Correct data type (discrete, continuous)

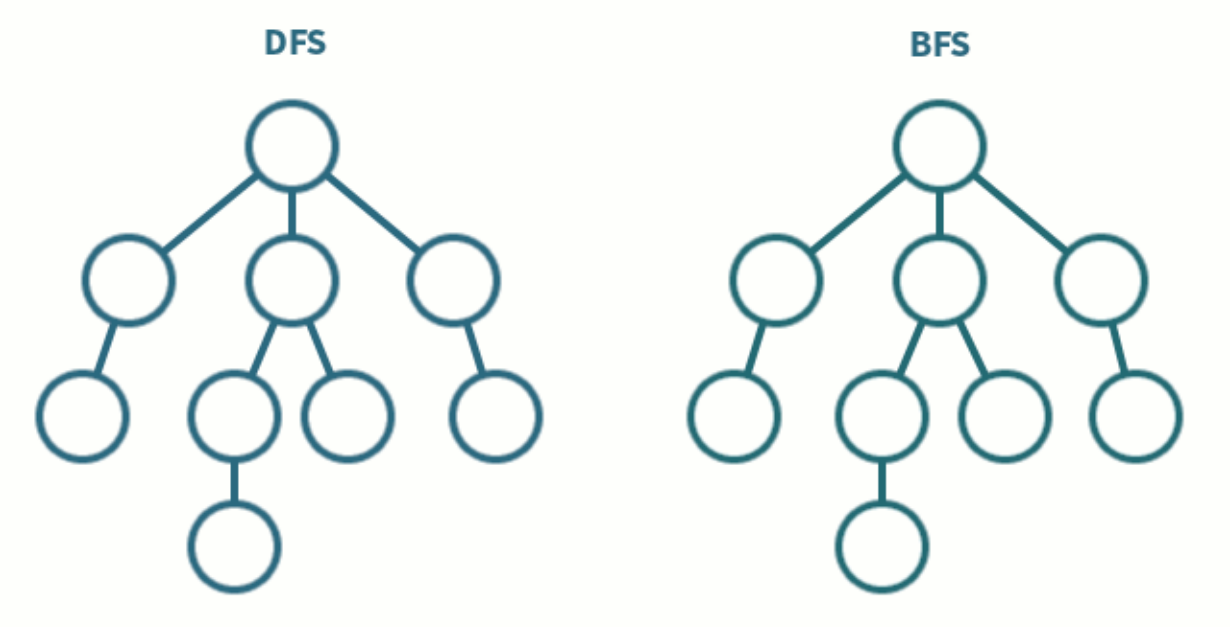
```
int discrete_action[3] = {-1, 0, 1};
float continuous_action = 0.95;
```

Search

- Un-informed **Blind Search**
  - Breadth-First Search (BFS): Uniform Cost search
  - Depth-First Search (DFS):
    - Depth-Limited Search (DLS)
    - Iterative-Deepening Search (IDS)

- **Informed Search**
  - Greedy Search: Heuristic
  - A* Search: Heuristic and path cost

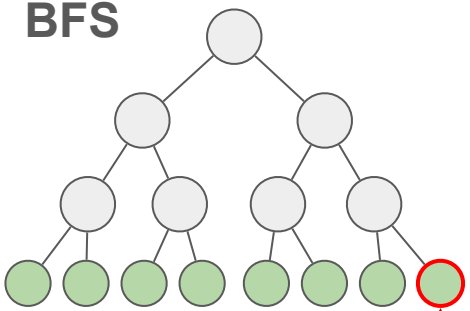
Search: Blind Search (BFS, DFS)



Search: Blind Search (BFS, DFS)

b: branching factor
d, **m**: max depth (goal depth)

BFS

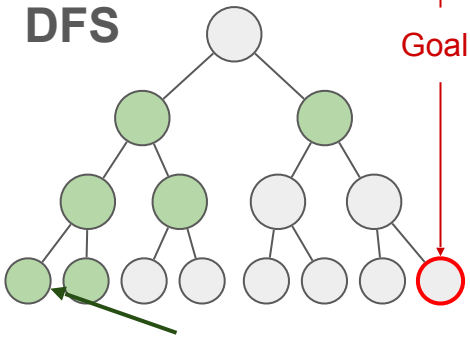


Complete	Yes
Optimal	Yes, if all step costs are equal
Time Complexity	$1 + b + b^2 + b^3 + \ldots + b^d = O(b^d)$
Space Complexity	$O(b^d)$

Worst case

2^3

DFS

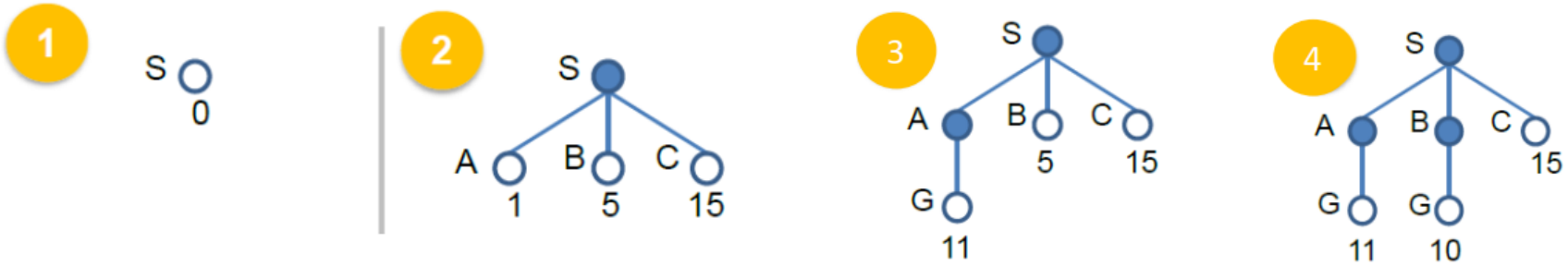


Complete	• infinite-depth spaces: No • finite-depth spaces with loops: No • with repeated-state checking: Yes • finite-depth spaces without loops: Yes
Time	$O(b^m)$ If solutions are dense, may be much faster than breadth-first
Space	$O(bm)$
Optimal	No

Worst case

2*3

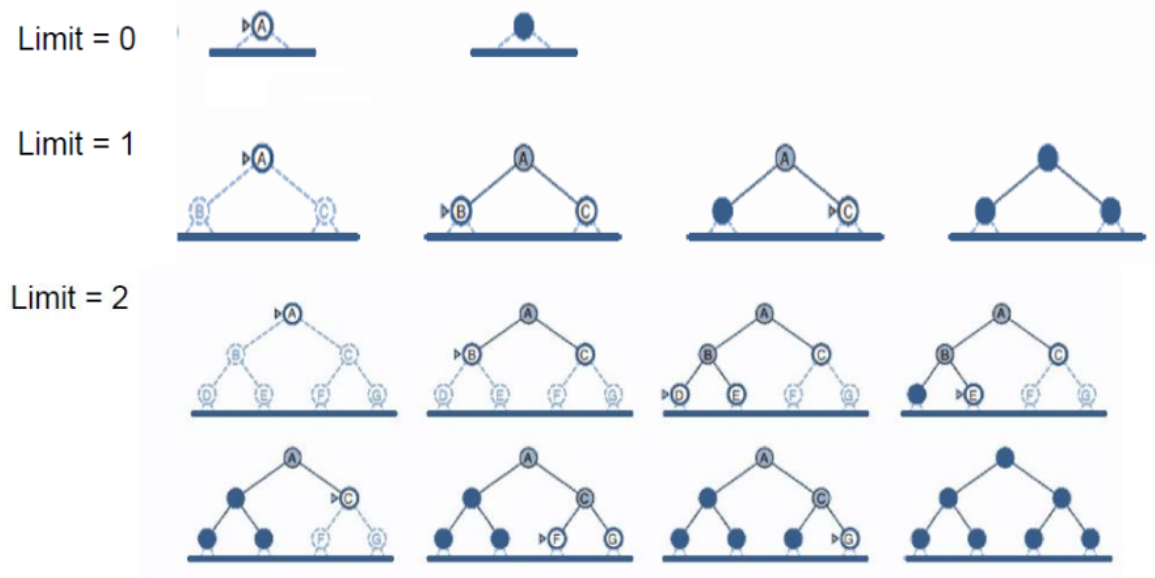
Search: Blind Search (Uniform Cost)



- Weighted graph
- Depth-first search until the depth is not as good as the next best breadth
- Keep track of the path cost

Note: Sometimes they give weighted graph, but ask for PURE BFS, DFS

Search: DLS, IDS



Criterion	Breadth-first	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Time	b^d	b^d	b^m	b^l	b^d
Space	b^d	b^d	bm	bl	bd
Optimal	Yes	Yes	No	No	Yes
Complete	Yes	Yes	No	Yes, if $l \geq d$	Yes

b: branching factor
d, **m**: max depth (goal depth)
l: limited depth

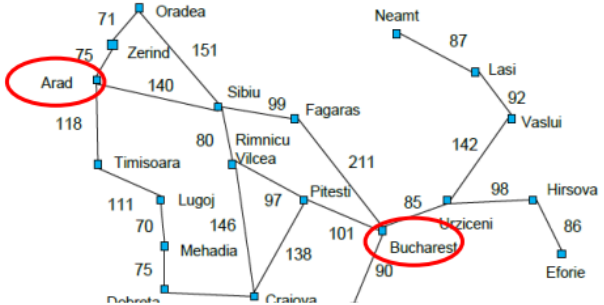
Search: Informed Search

- Blind search expands the search from the **source** in all directions
  - Only know it is the goal when it reaches **the goal**
- Informed Search has another information to measure the “distance/ farness” **from source to the goal**

- Keep track of the “desirability/score/utility” of each node
  - Uniform-cost Search: Path cost only
  - Greedy Search: Heuristics only
  - A* Search: Heuristics + Path

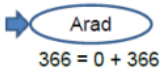
Search: Informed Search

- Hybrid Evaluation f(n) = g(n) + h(n)

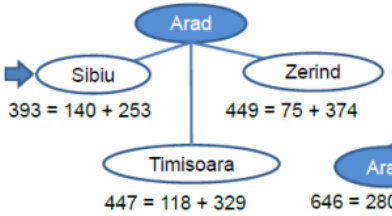


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Efoire	161
Fagaras	176
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

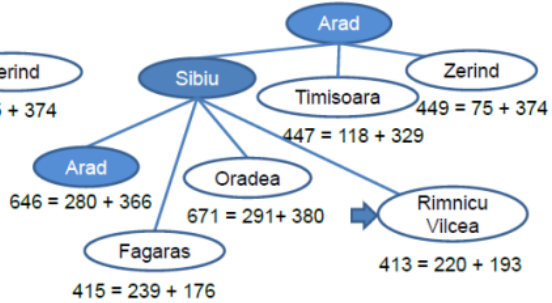
(a) The initial state



(b) After expanding Arad

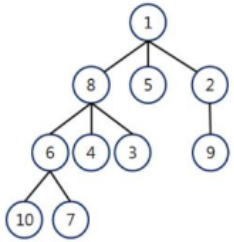


(c) After expanding Sibiu



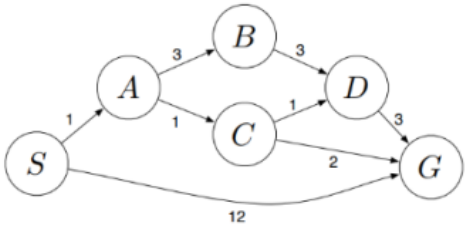
Question

9. Which node does Breadth First Search select to expand after nodes 1, 8, and 5 (the number in each circle means the ID of a node)?



a) Node 6 b) Node 4 c) Node 3
d) Node 2

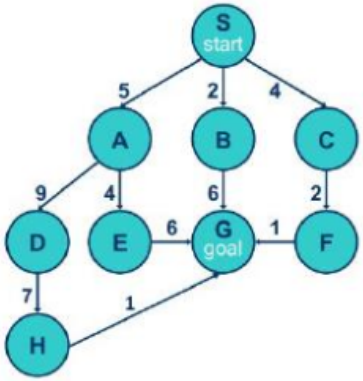
10. What path would Breadth-First Search return for this search problem (initial state is S and the goal state is G)?



a) S,G b) S, A, C, G c) S,A,B,D, G
d) S, A, C, D,G

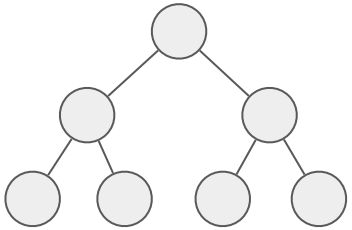
Question

What path would Depth-First Search return for this search problem (initial state is S and the goal state is G, searching from left to right)?



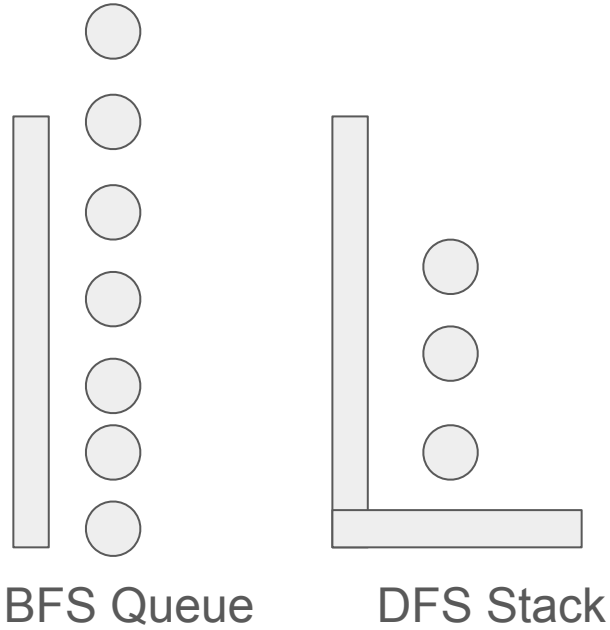
a) S, A, E, G
b) S, B, G
c) S, C, F, G
d) S, A, D, H, G

Search: Graph (Data Structure & Algo)



- Graph
  - Root node, Parent nodes, Children node
- Order
  - First-in-First-Out Queue
  - Last-in-First-Out Stack
- Keep track of the nodes values with arrays

- Shortest-path from node A to B in a known weighted graph: Dijkstra’s Algo

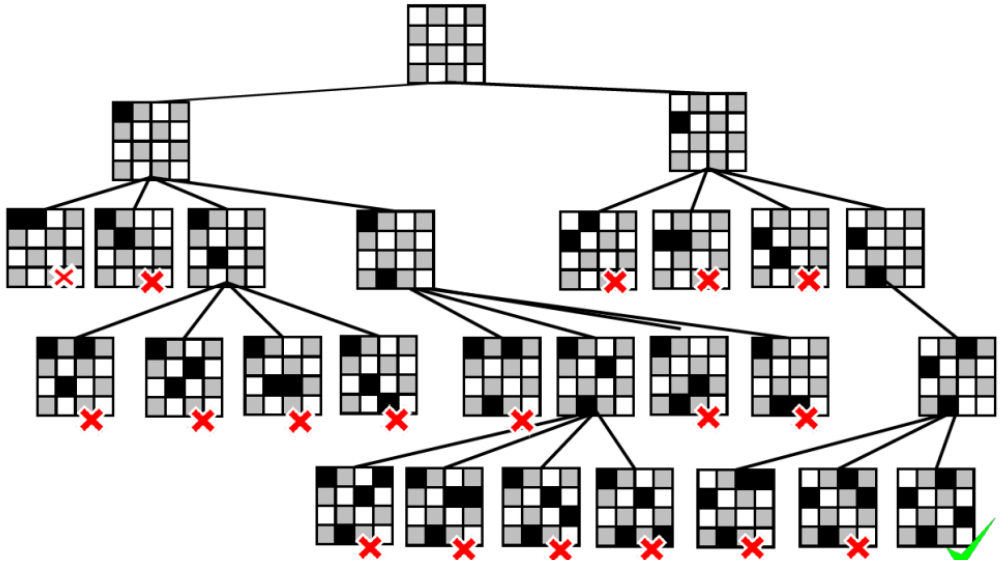


Constraint Satisfaction

- Backtracking:
  - Depth first search
  - With constraint checking
- Forward Checking,
- Constraint Propagation

Terminology:
- **Variables**
- Variables have **values**

Objective:
- Find a set of values for variables that satisfied constraints
- **Need not be unique**

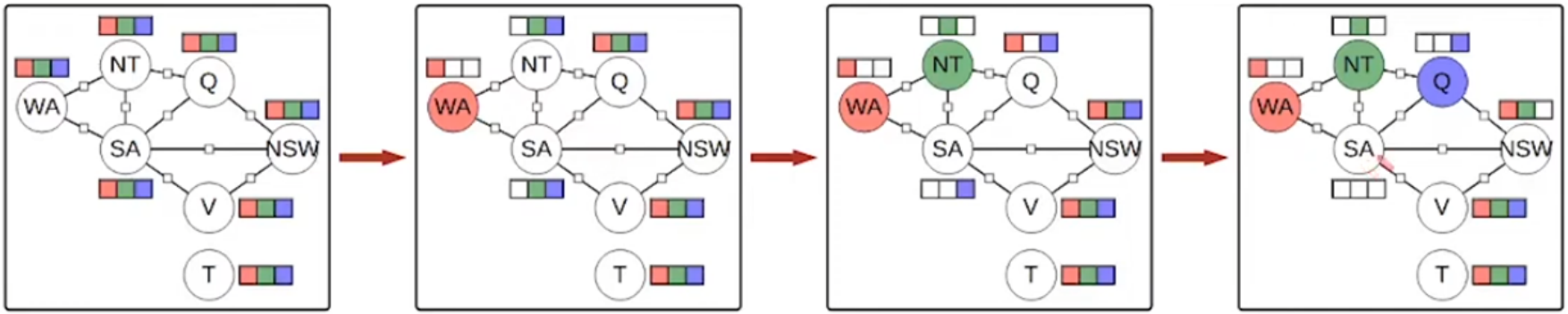


Constraint Satisfaction

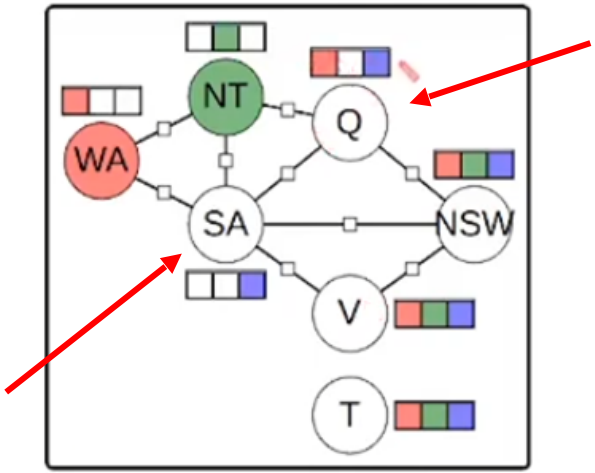
Terminology:
- **Variables**
- Variables have **values**

Objective:
- Find a set of values for variables that satisfied constraints
- **Need not be unique**

- Backtracking:
  - Depth first search
  - With constraint checking
- Forward Checking,
- Constraint Propagation



Constraint Satisfaction



Which **variable** to allocate first?

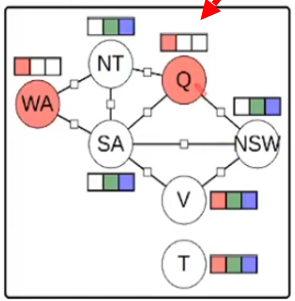
Key idea: most constrained variable

Choose variable that has the smallest domain.

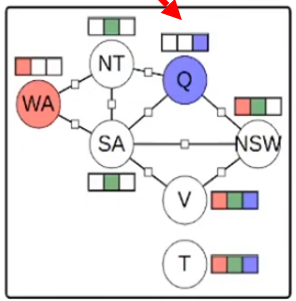
Constraint Satisfaction

Which **value** to allocate for Q first?

What values to try for Q?



$2 + 2 + 2 = 6$ consistent values



$1 + 1 + 2 = 4$ consistent values

Key idea: least constrained value

Question

A Constraint Satisfaction Problems (CSP) consists of a set of variables to which values can be assigned such that a set of constraints over the variables is respected.

a)True b)False

12. Which is the most constrained variable (the variable that is involved in the largest number of constraints) in the map-coloring problem (The terms in the figure are the names of variables)?

a) Northwest

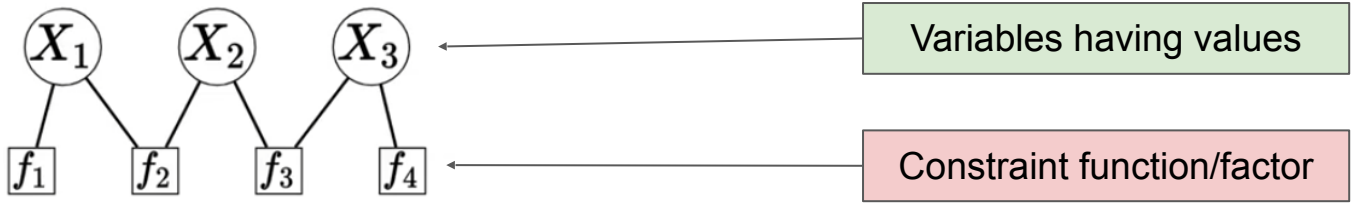
b) Central Singapore

c) Northeast

d) Southwest

Constraint Satisfaction

- Factor Graph



x_1	$f_1(x_1)$
R	0
B	1

x_1	x_2	$f_2(x_1, x_2)$
R	R	1
R	B	0
B	R	0
B	B	1

x_2	x_3	$f_3(x_2, x_3)$
R	R	3
R	B	2
B	R	2
B	B	3

x_3	$f_4(x_3)$
R	2
B	1

$f_1(x_1) = [x_1 = \text{B}]$ $f_2(x_1, x_2) = [x_1 = x_2]$ $f_3(x_2, x_3) = [x_2 = x_3] + 2$ $f_4(x_3) = [x_3 = \text{R}] + 1$

Game Playing: Minimax

- Have a function to calculate the **utility/score** at the current state of the game
- The utility score favor **us**

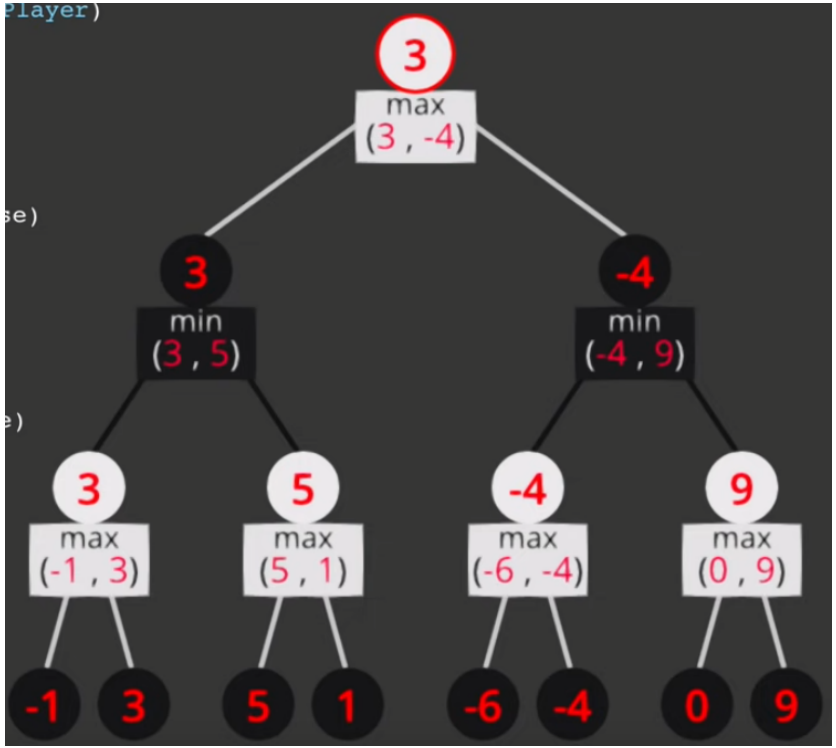
- We want to **maximize**, the opponent want to **minimize**

- Propagate **UP** from the bottom nodes

- Challenges: needs to evaluate all the way to the endgame. Not feasible!

https://www.youtube.com/watch?v=l-hh51ncgDI

Game Playing: Minimax

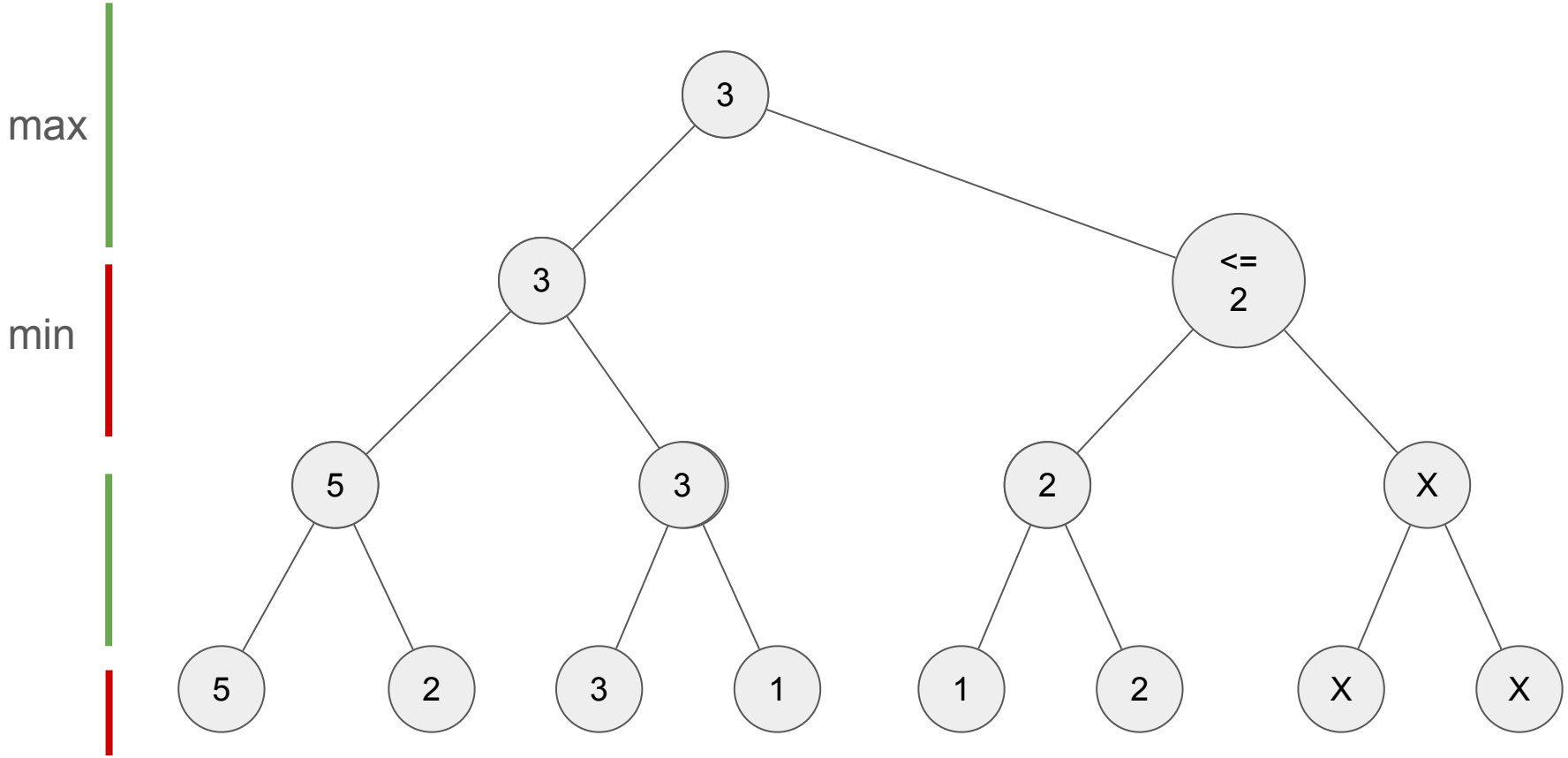


You can calculate the “score” of any state

You can look ahead some step and know their utility

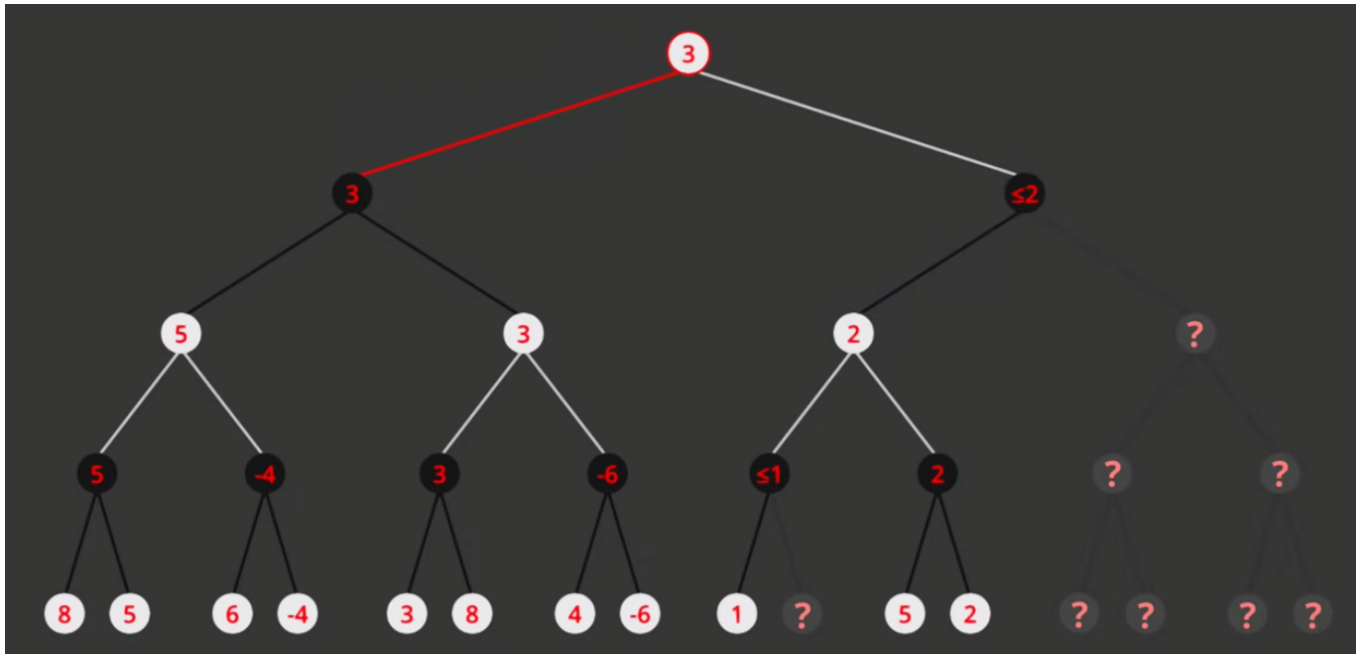
What is your current utility? What should you do?

Game Playing: Minimax



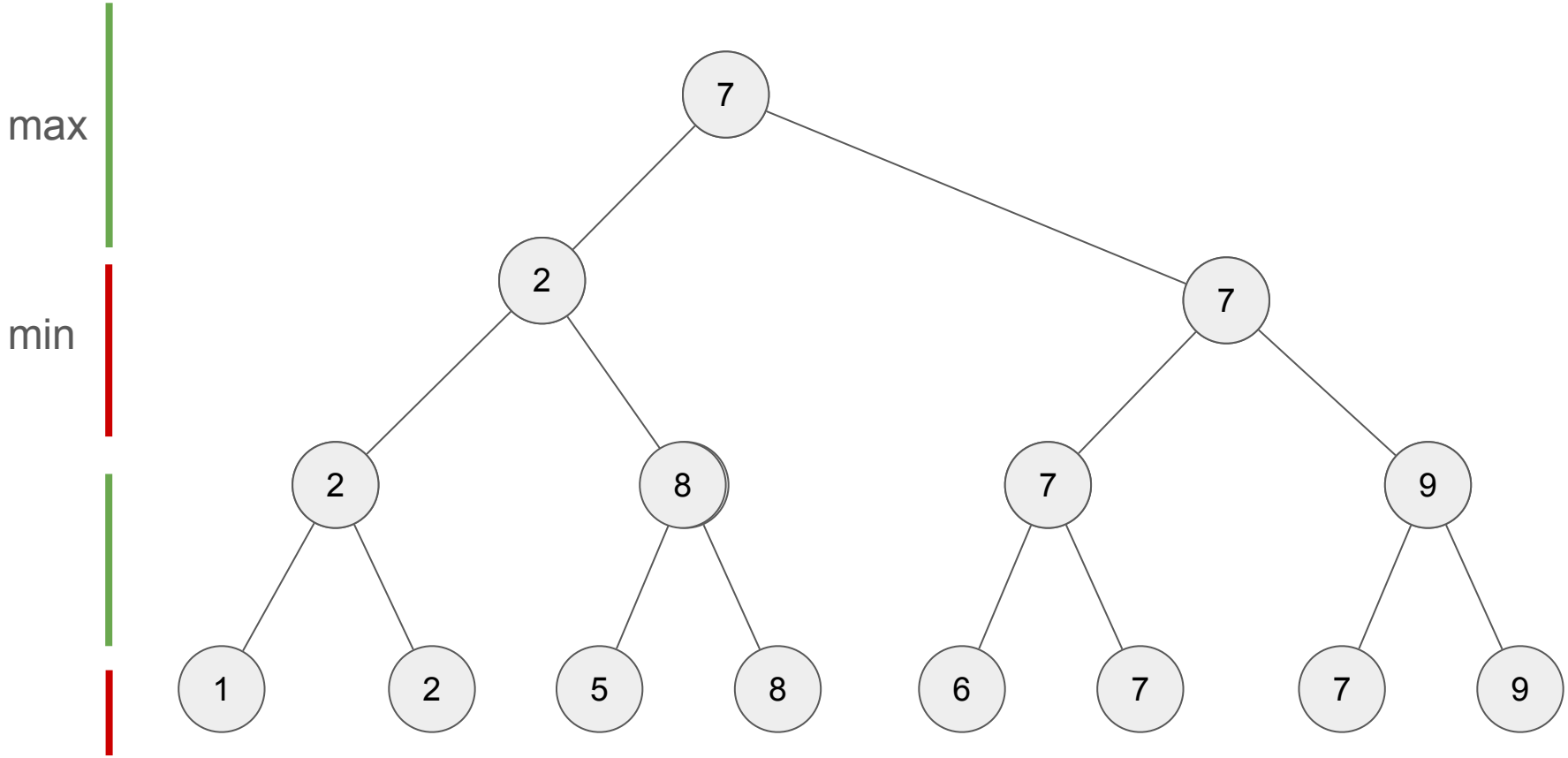
Game Playing: Pruning

- Agent does not need to evaluate all possibilities. Can use pruning



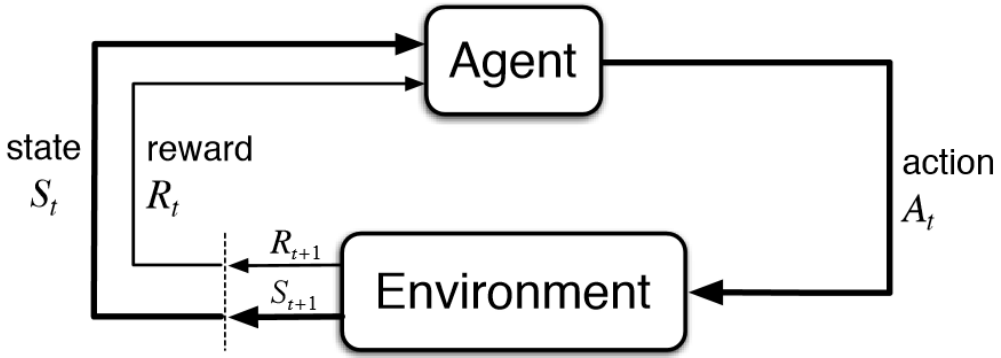
https://www.youtube.com/watch?v=l-hh51ncgDI

Game Playing: Minimax



Reinforcement Learning

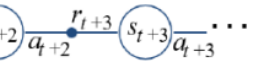
- State, Action, Reward
- Expected Utility
- Utility, Bellman Equation

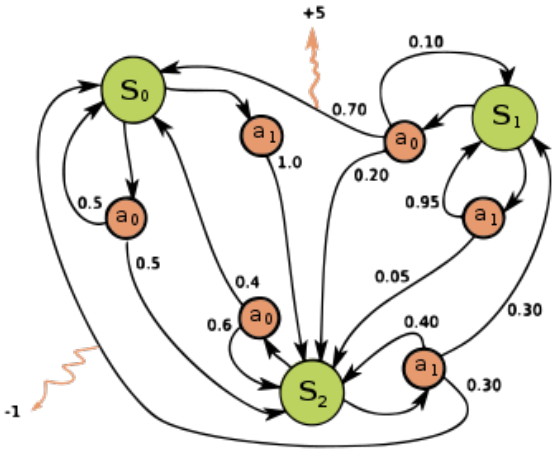


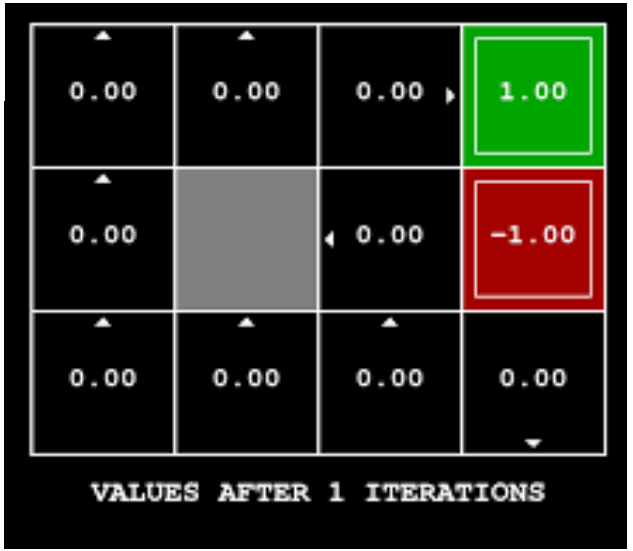
OBJECTIVE
Maximize the reward received as the agent operate in the environment (from start to end/goal state)

Markov Decision Process

- MDP components:
  - **States** s
  - **Actions** a
  - **Transition model** P(s' | s, a)
  - **Reward function** R(s)
- The solution:
  - **Policy** π(s): mapping from states to actions







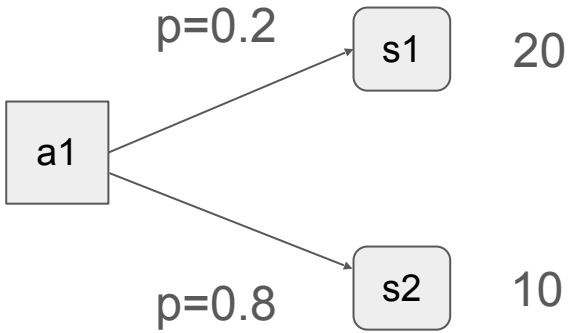
https://www.youtube.com/watch?app=desktop&v=OvTEhNL96v0

Expected Utility

- Recall your statistics and probability
  - It is just calculating the “Mean”, “Expected Value”

x	0	1	2
$p(x)$	0.16	0.48	0.36

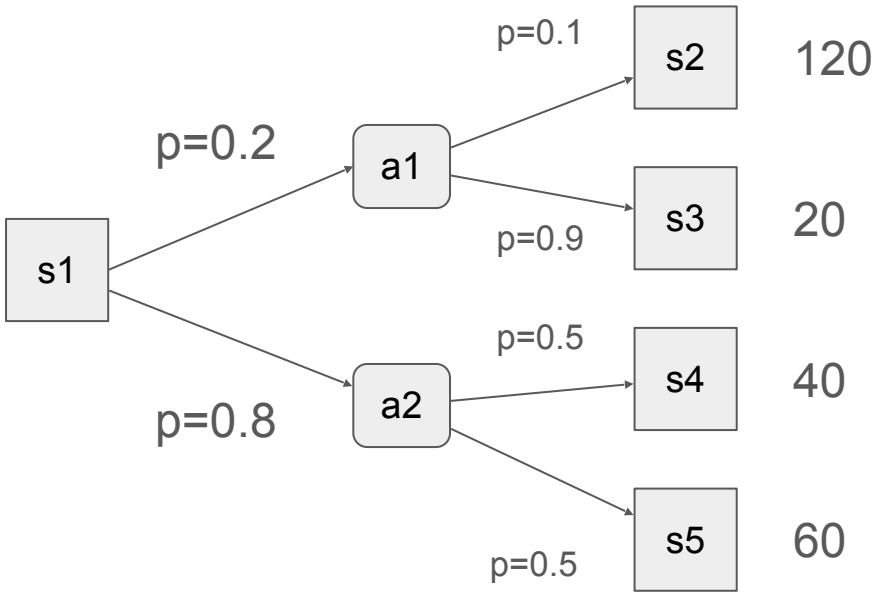
$$E(X) = \sum_{\text{all } x} xp(x)$$
$$= 0 \cdot 0.16 + 1 \cdot 0.48 + 2 \cdot 0.36$$



Why the distribution? Non-deterministic, probabilistic action and state transitions.

Expected utility of action “a1” = 8+4 = 12

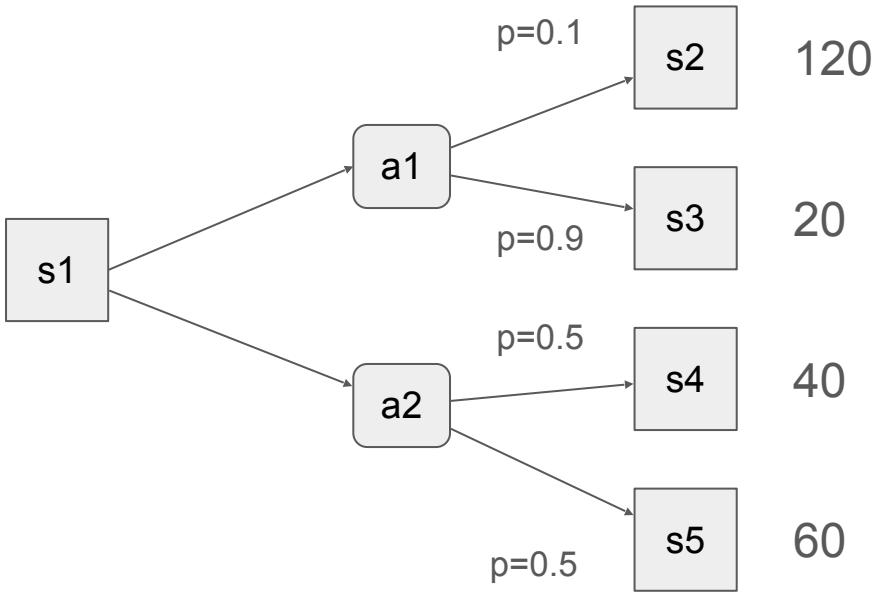
More calculation



Probabilistic
Action policy

Expected utility of state “s1”
= 0.2 (12+18) + 0.8 (20+30)
= 0.2 * 30 + 0.8 * 50
= 6 + 40
= 46

More calculation



a1 vs a2
(12+18) vs (20+30)
30 vs 50

Deterministic policy
Choose the better action

Discounted Utility

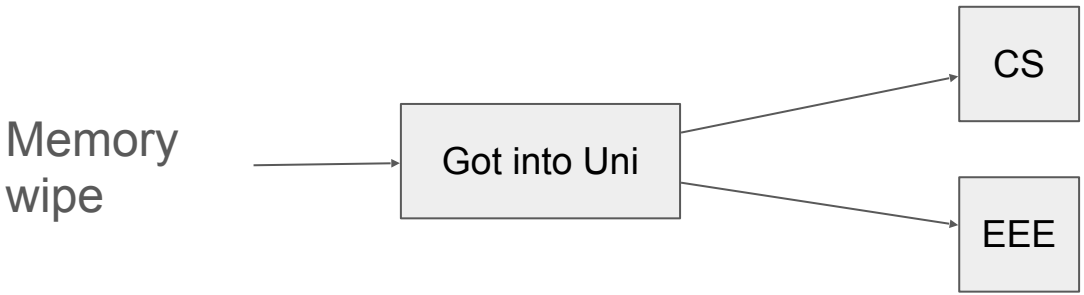
Solution: *discount* the individual state rewards by a factor γ between 0 and 1:

$$U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$$
$$= \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \frac{R_{\max}}{1-\gamma} \qquad (0 < \gamma < 1)$$

- γ^t will decrease to 0 as t increase, due to the constraint $0 < \gamma < 1$
- Even for infinite state sequences, the total utility is bounded.

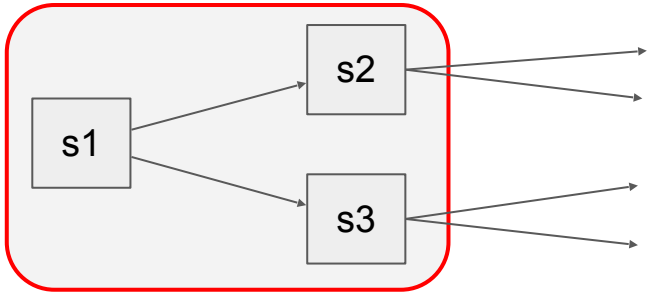
Intuitive idea of **Utility**

- **No history of previous states**
- If you reach the same state, you will see the same “state”, same set of actions, have the same set of transition probability
- Question: If you are to redo your University, you don’t remember anything that you have done, what would you choose?



Intuitive idea of **Utility**

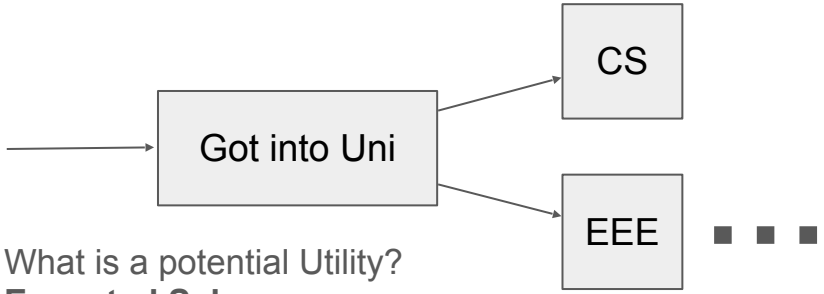
- The **Utility encapsulates the "desirability" toward the goal.**
- Agent can only observe the current and the possible transitions to the next state



What the agent can observe. How to know which state to go?

Goal

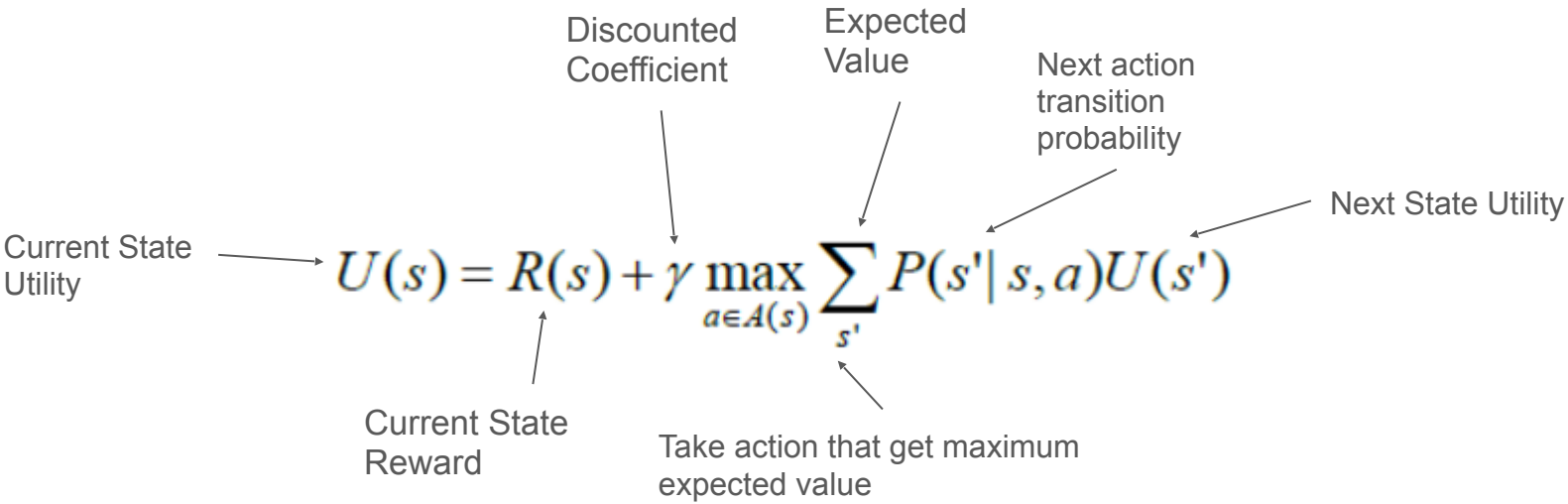
Memory wipe



What is a potential Utility?
Expected Salary

Goal: Get money

Bellman Equation



Bellman Equation

Discounted Coefficient

Expected Value

Next action transition probability

Next State Utility

Current State Utility

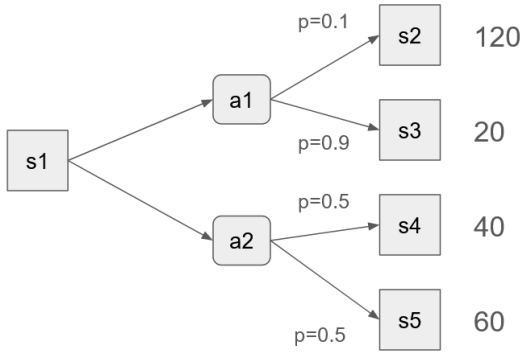
$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$

Current State Reward

Take action that get maximum expected value

The point?
Propagate the utility score to states that are further away from the goal

Why take max of expected value?
Probabilistic state transition. You want to take an action, but life hits you in the face
Take max because we want to reach the goal, have the best utility



10

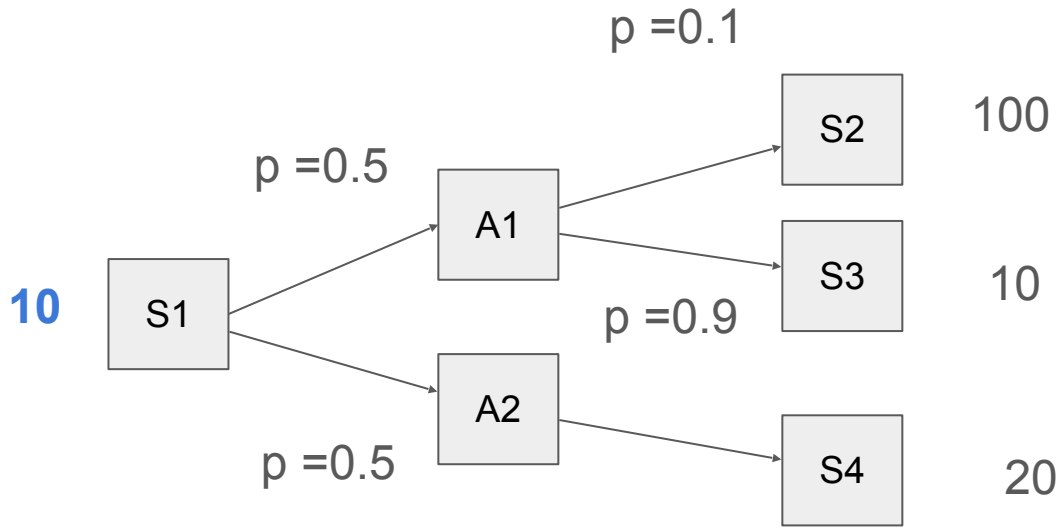
$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

5

A2 = 10

Gamma = 0.5
Reward = 5

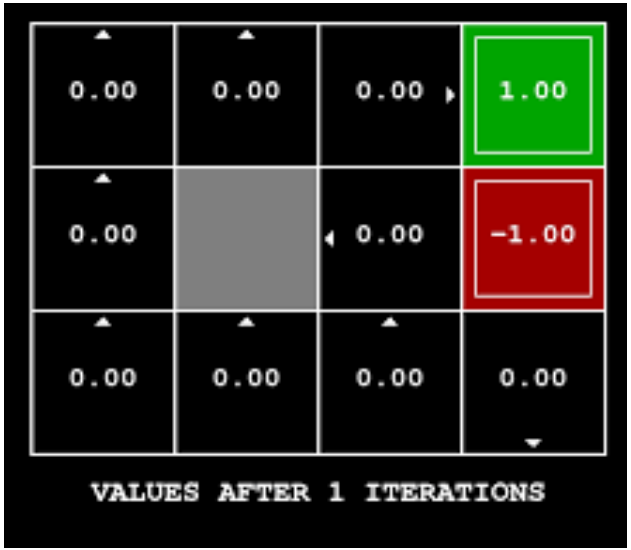
A1
= Pa1 * Ua1 = 0.5*19
= 9.5

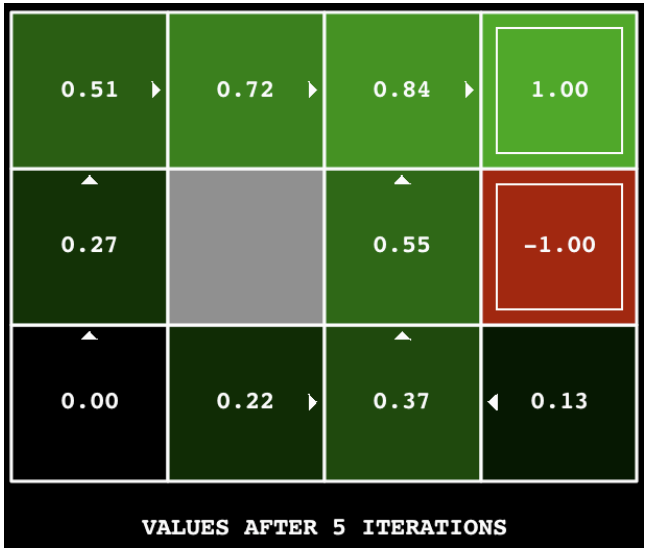


A2
= Pa2 * Ua2
= 0.5 * 20
= 10

Value Iteration

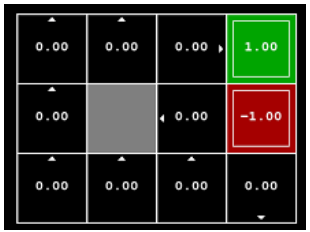
- Greedy Policy
  - Evaluate the Utility of all state
  - Pick the state with highest utility



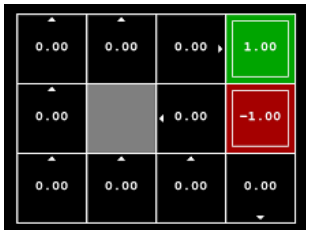


Over time

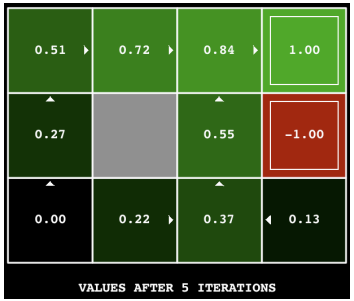
1 run/episode from start to end condition (might not be the goal)



1 run/trajectory



1 run/trajectory



“time”, “time step”

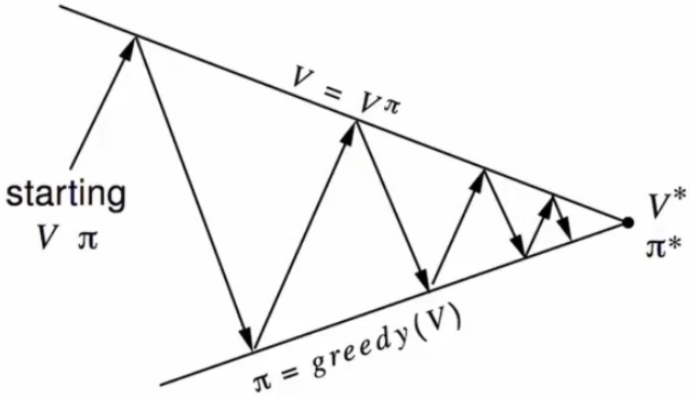
Policy Iteration

- Agent has its own policy
  - Observing different states, the agent will give different action
- Evaluate the current policy
- Update the policy
  - Can be a look up table
  - Can be a function, mapping, neural network

- Alternating between the following two steps

  - **Policy evaluation:** calculate $U^{\pi_i}(s)$ for every state s
  - **Policy improvement:** calculate a new policy π_{i+1} based on the updated utilities

$$\pi^{i+1}(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s' | s, a) U^{\pi_i}(s')$$



Policy evaluation Estimate v^{π} / q^{π}

Policy improvement Generate $\pi' \geq \pi$