Artificial Intelligence in Video Games

Kaleb, David, Rodney, Trevor

Presentation Overview

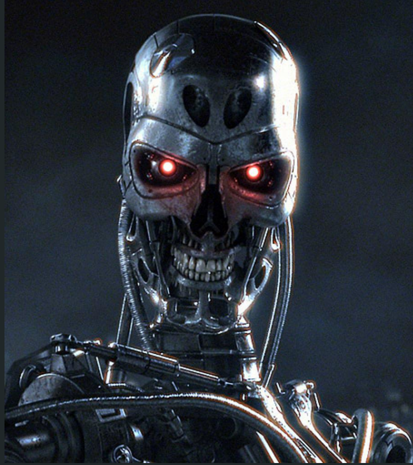
- General Overview of AI
  - What’s AI?
  - Why use AI for Games?
  - Chess - Decision Tree Navigation
  - FEAR - Finite State Machines
- Deterministic Search Algorithms
  - What is a deterministic search?
  - What are they used for?
  - Uninformed vs Informed
  - Examples
- Search Algorithm
  - Graph Theory
  - Search Algorithms
  - CSP
  - Adversarial Search Algorithms
- Machine Learning
  - What is machine learning?
  - Deep Learning
  - Machine Learning to Play Games
  - AlphaStar and DeepMindAI

AI - Overview and Examples

Kaleb Johnson

Artificial Intelligence

- Pop culture icons
  - Bogeymen: Terminator, 2001
  - Examination of ourselves: Ex Machina, Asimov, Frankenstein

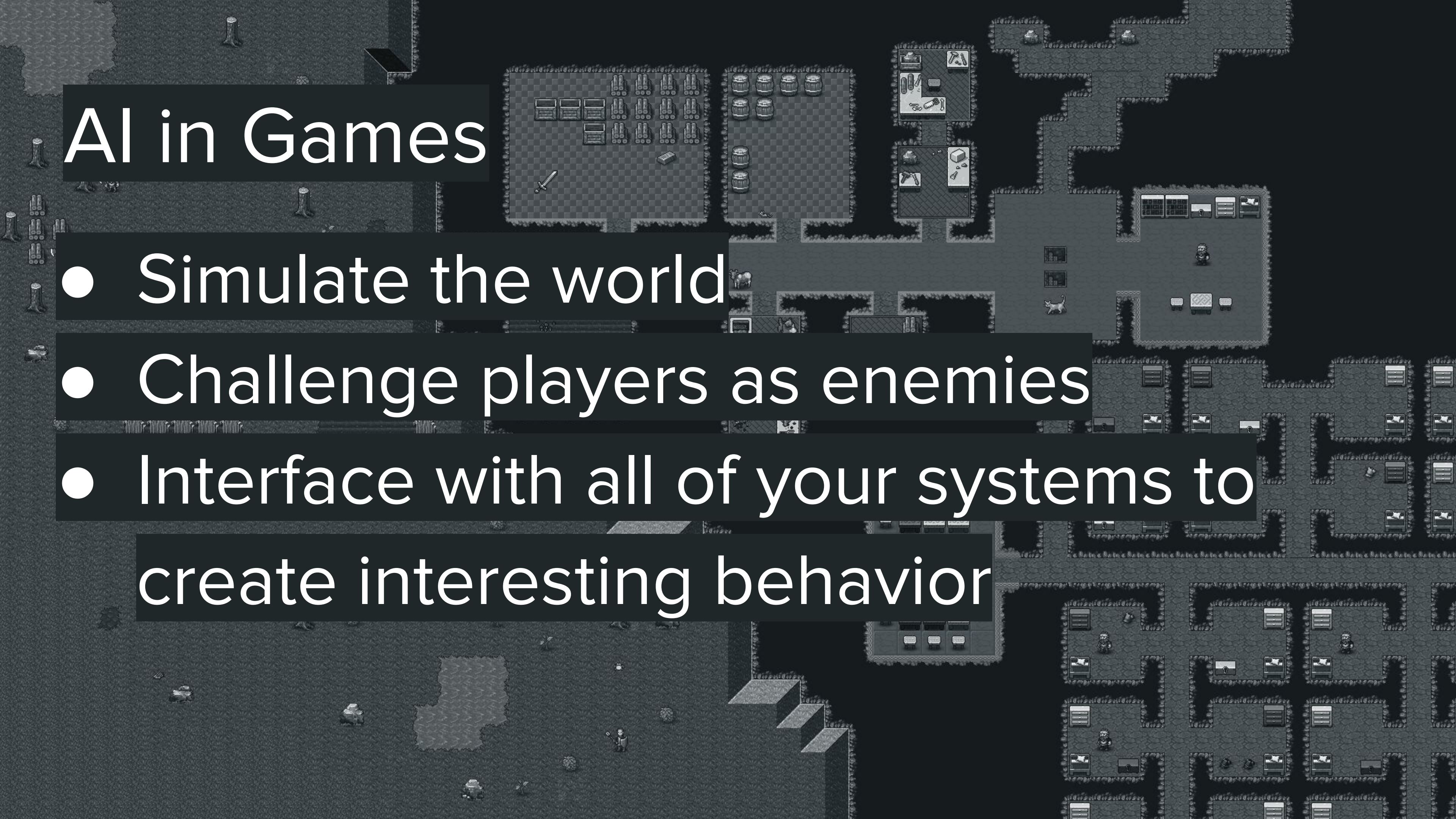


Artificial Intelligence

- Ubiquitous part of the modern world.
  - Content delivery algorithms
  - Data analysis - Makes free services profitable
  - So many other tools

Artificial Intelligence

- Central to Most Video Games
  - Core to single-player interactions
  - Provides tools for multi-player experiences



AI in Games

- Simulate the world
- Challenge players as enemies
- Interface with all of your systems to create interesting behavior

Two Examples

- Chess
  - Classical Board Game
  - Most well-documented game AI
- F.E.A.R.
  - 2005 First Person Shooter
  - Legendary for enemy AI

Chess AI

- Easily mathematically represented
  - Perfect Information
- Evaluations are easy to implement
- Long history and multitude of approaches

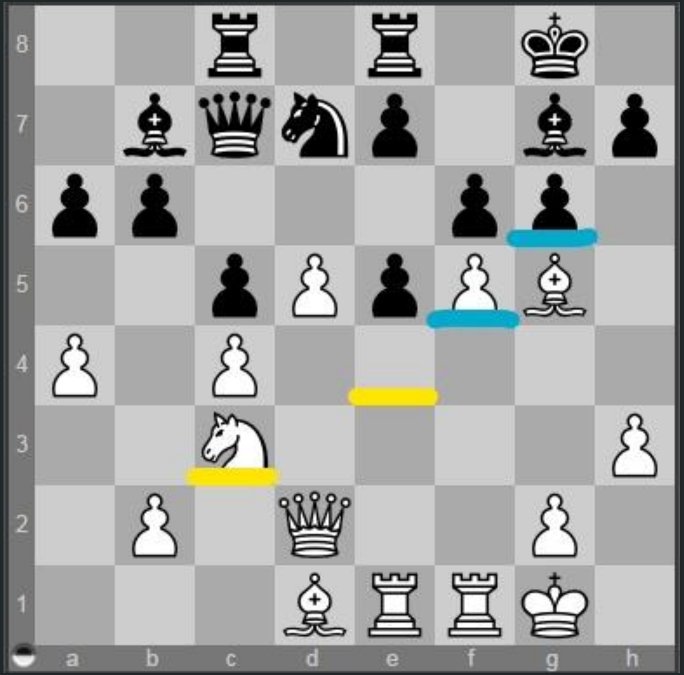
Chess - Decision Trees

- Evaluate a board
    - Checkmate
    - Piece Point Evaluation
    - Further Methods
        - Pawn Structure, Trapped Pieces, King Weakness, Center Control, etc.

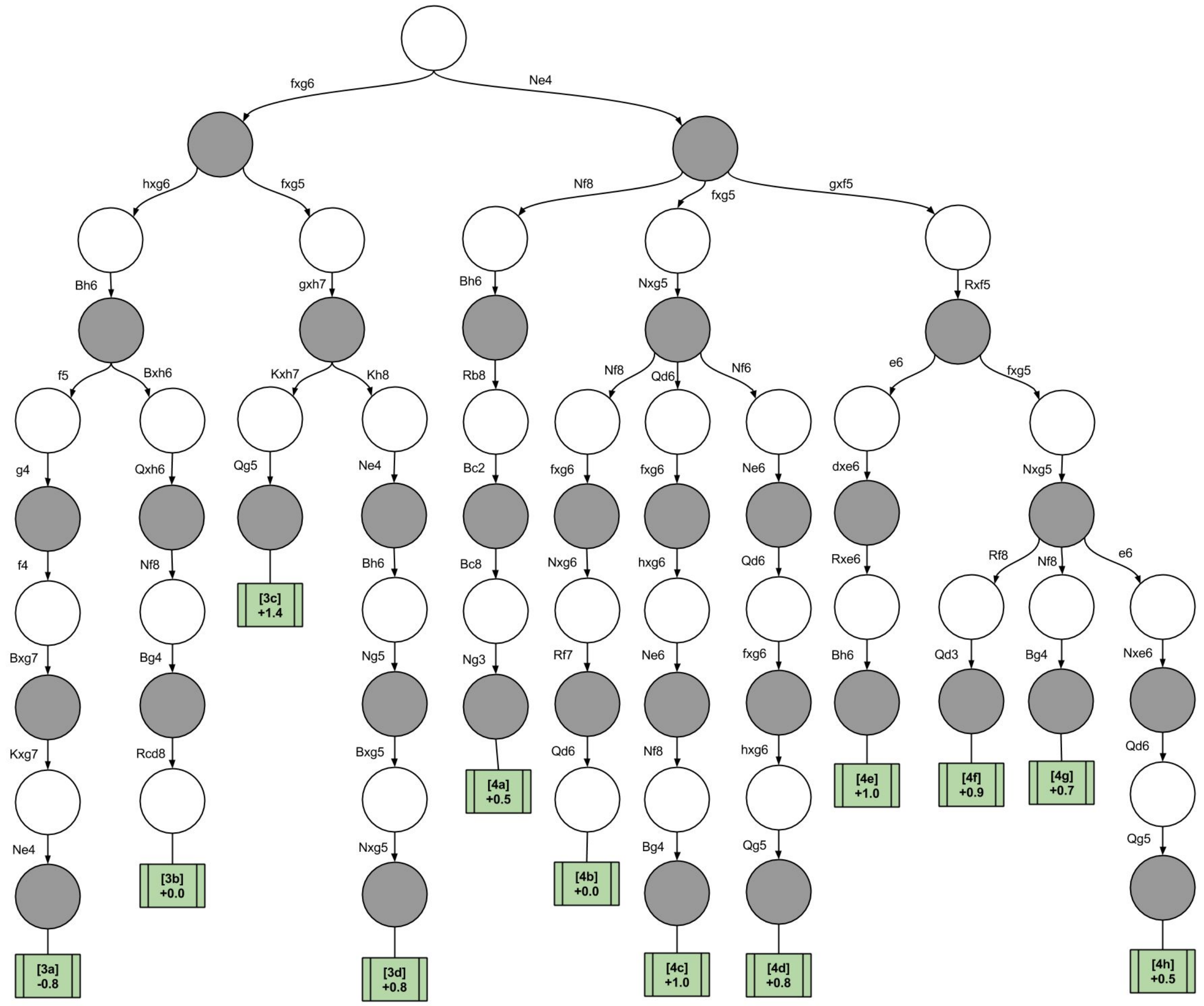
Chess - Decision Trees (2)

- Analyze possible moves
    - Evaluate the board at the new positions
    - Evaluate the board after the opponent’s move

Chess - Decision Tree Example (1)



Chess - Decision Tree Example(2)



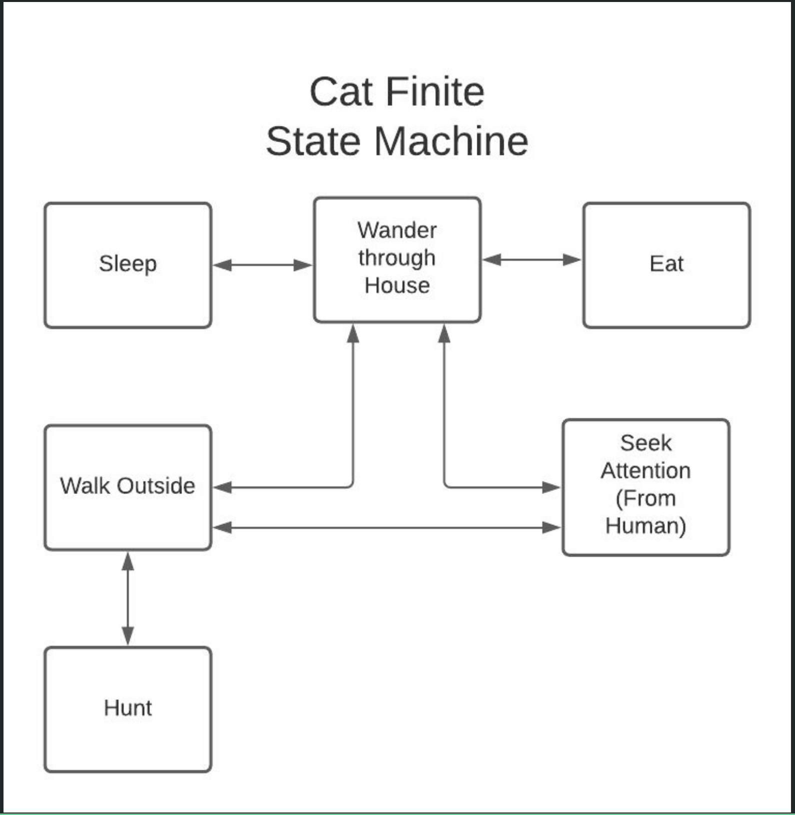
Chess - Exceptions

- Opening Books
- DeepMind and AlphaGo

F.E.A.R.

- 2005 First Person Shooter
- Legendary AI
- Simple techniques
    - FSM
    - A*
    - GOAP

F.E.A.R. - Finite State Machines



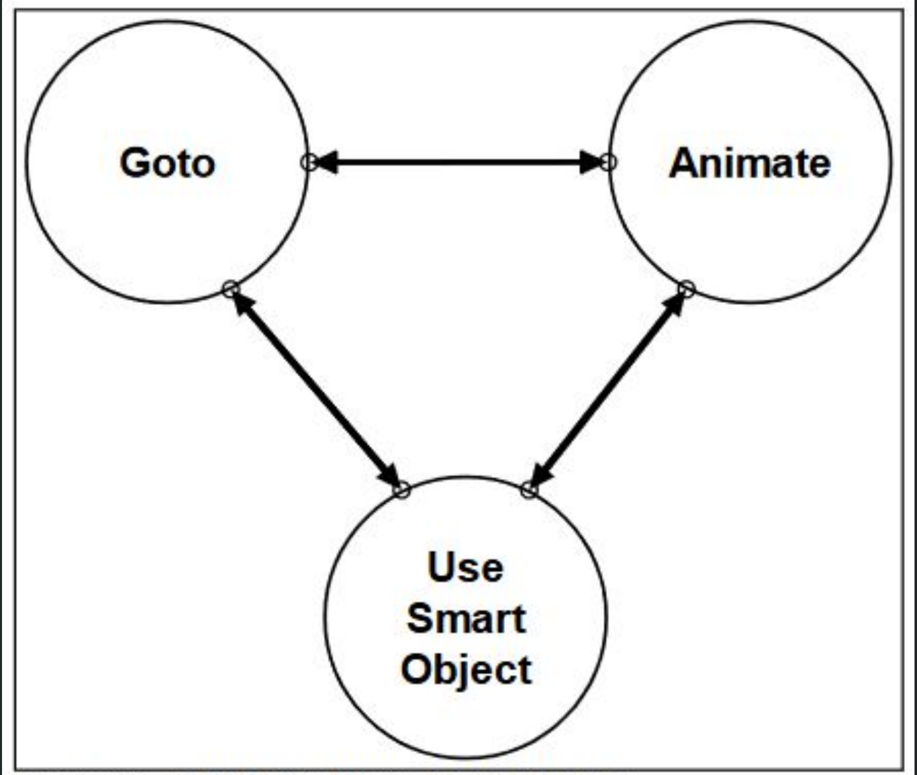
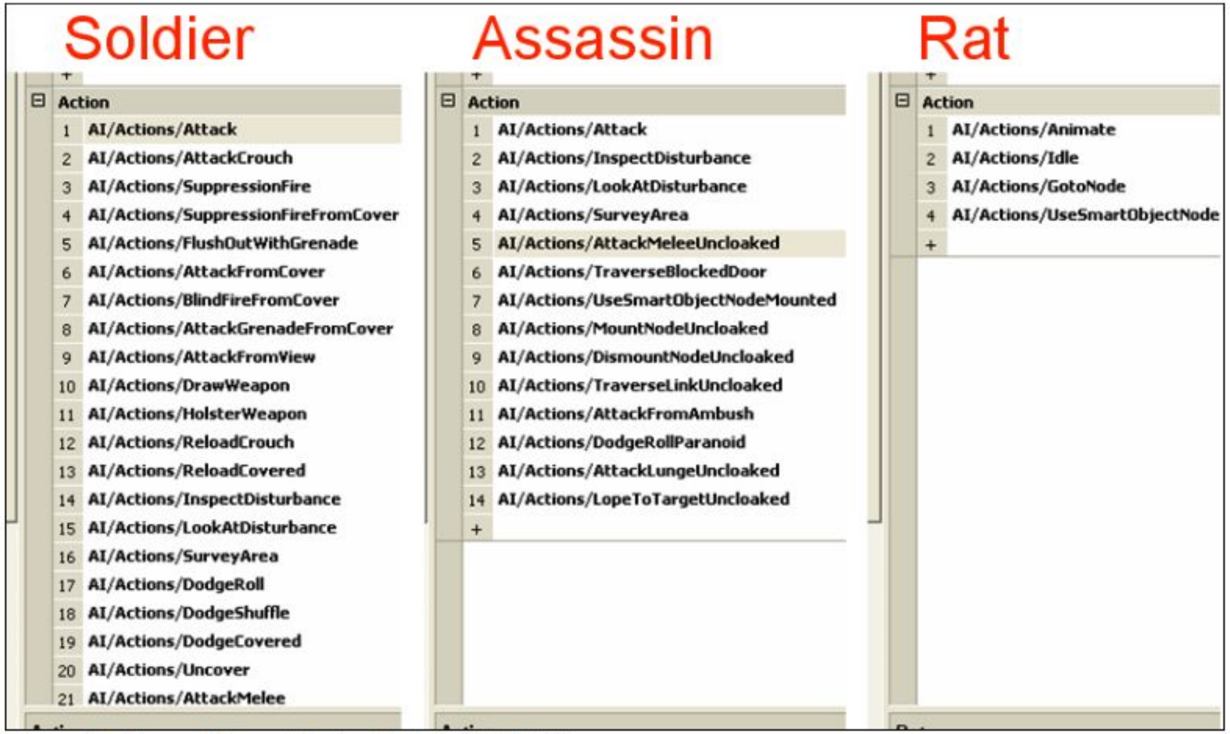


Figure 2: *F.E.A.R.*'s Finite State Machine

F.E.A.R. - Planning System

- STRIPS/GOAP are Planning Systems
  - Stanford Research Institute Problem Solver
  - Goal Oriented Action Planning
- Entities have goals and actions

F.E.A.R. - GOAP Example Actions



Soldier

	Action
1	AI/Actions/Attack
2	AI/Actions/AttackCrouch
3	AI/Actions/SuppressionFire
4	AI/Actions/SuppressionFireFromCover
5	AI/Actions/FlushOutWithGrenade
6	AI/Actions/AttackFromCover
7	AI/Actions/BlindFireFromCover
8	AI/Actions/AttackGrenadeFromCover
9	AI/Actions/AttackFromView
10	AI/Actions/DrawWeapon
11	AI/Actions/HolsterWeapon
12	AI/Actions/ReloadCrouch
13	AI/Actions/ReloadCovered
14	AI/Actions/InspectDisturbance
15	AI/Actions/LookAtDisturbance
16	AI/Actions/SurveyArea
17	AI/Actions/DodgeRoll
18	AI/Actions/DodgeShuffle
19	AI/Actions/DodgeCovered
20	AI/Actions/Uncover
21	AI/Actions/AttackMelee

Assassin

	Action
1	AI/Actions/Attack
2	AI/Actions/InspectDisturbance
3	AI/Actions/LookAtDisturbance
4	AI/Actions/SurveyArea
5	AI/Actions/AttackMeleeUncloaked
6	AI/Actions/TraverseBlockedDoor
7	AI/Actions/UseSmartObjectNodeMounted
8	AI/Actions/MountNodeUncloaked
9	AI/Actions/DismountNodeUncloaked
10	AI/Actions/TraverseLinkUncloaked
11	AI/Actions/AttackFromAmbush
12	AI/Actions/DodgeRollParanoid
13	AI/Actions/AttackLungeUncloaked
14	AI/Actions/LopeToTargetUncloaked
+	

Rat

	Action
1	AI/Actions/Animate
2	AI/Actions/Idle
3	AI/Actions/GotoNode
4	AI/Actions/UseSmartObjectNode
+	

Figure 4: Three different Action Sets in GDBEdit.

F.E.A.R. - Planning Hierarchies

- Goals are Ranked
- Actions have Costs
- A* is used to search the Decision Space

F.E.A.R. - Results of GOAP

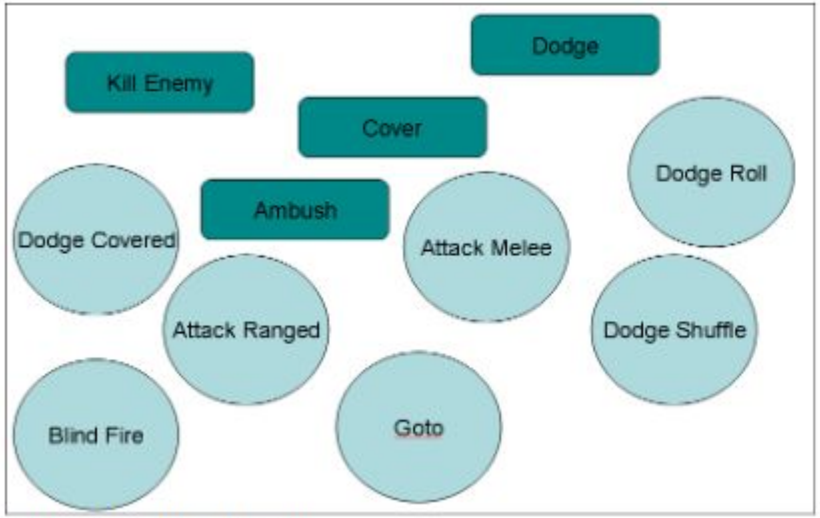
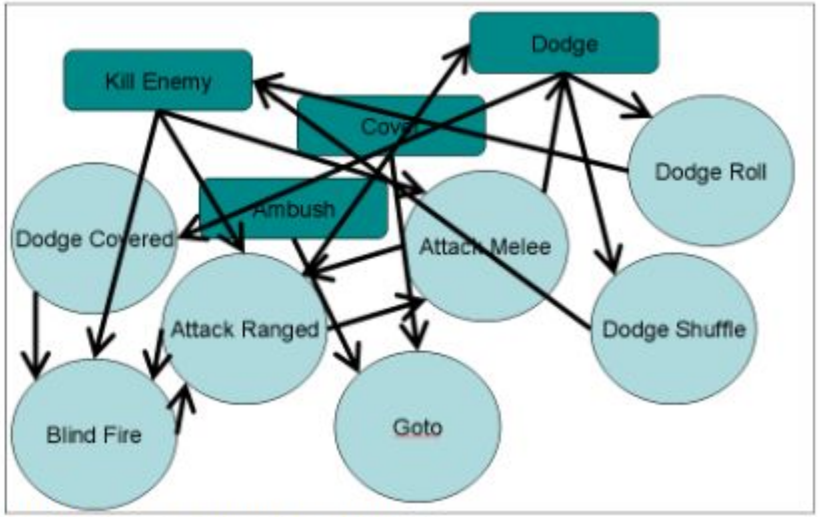


Figure 6: We do this.



But we never have to do this.

F.E.A.R. - Simple but Effective Design

- Not managing every interaction
- Fitting your AI to your game
- Tailoring information to players

Deterministic Search Algorithms

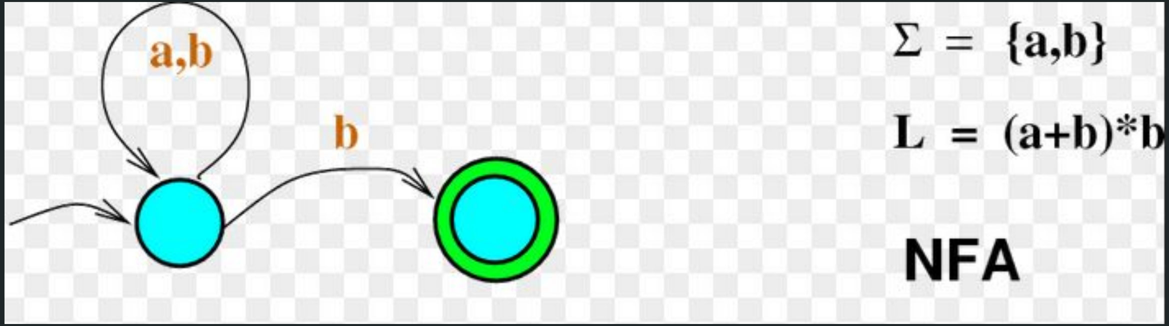
David Bush

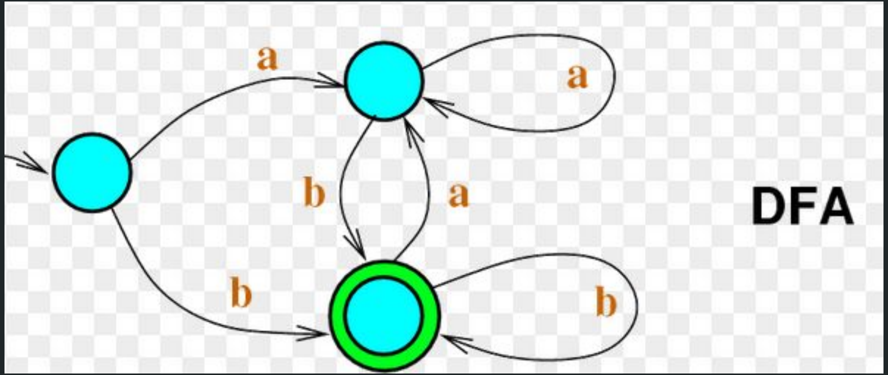
Deterministic vs Non-deterministic

- A deterministic algorithm can determine what the next step will be and will always have the same output even if the input remains the same.

- A non-deterministic algorithm can’t determine the next step because there are more than one path it could take. It will also give different outputs with different executions for the same input.

NFA vs DFA



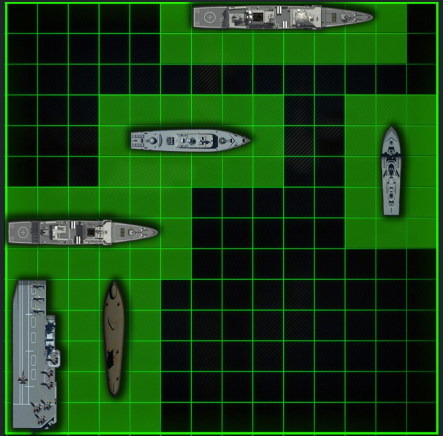


What is a Deterministic Search?

- A search that’s output won’t change given the same input.
- It also will find a solution if one exists. If there is no solution then it will have checked every possible one. Can be limited by the amount of time it has to search. (Complete search)
- Can determine what the next step in the search will be. (Not random)

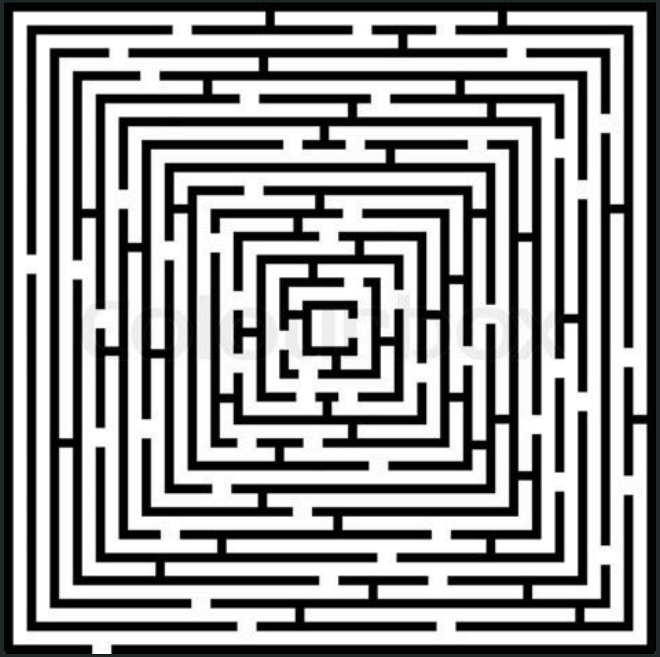
What games are deterministic?

- Imperfect information
  - Battleships, minesweeper
- Perfect information
  - Chess, checkers, connect4, Nim, tic-tac-toe



Deterministic Searches

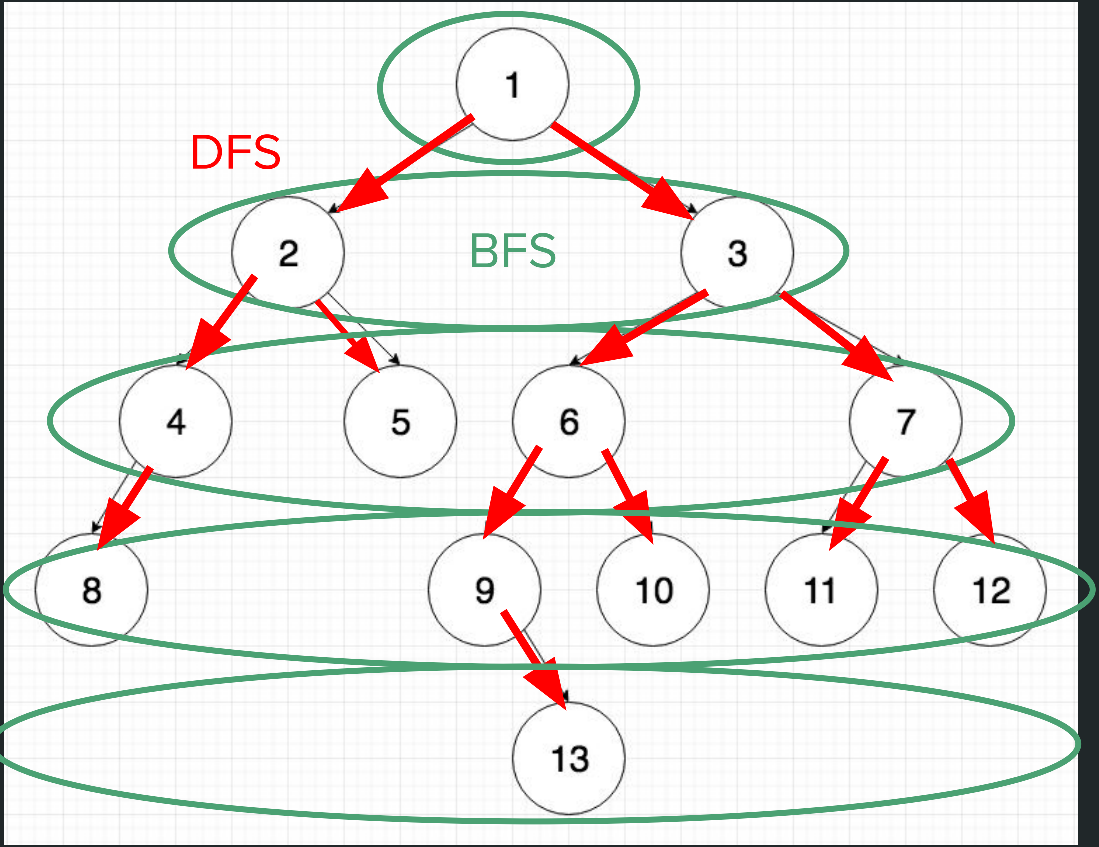
- Uninformed
  - Doesn’t have additional information, such as how far away the goal is.
- Informed
  - Has additional information like knowing how far away the goal currently is. (Heuristic)



Uninformed Searches

- Depth-first Search
  - Starts at the root and explores each branch as far as it can go before backtracking or a solution is found.
  - Not optimal
- Breadth-first Search
  - Starts at the tree root and explores all the nodes at a given level of the tree before moving to the next level.
  - Finds the optimal solution in the shortest number of actions/moves

Depth-first vs Breadth-first



Informed Searches

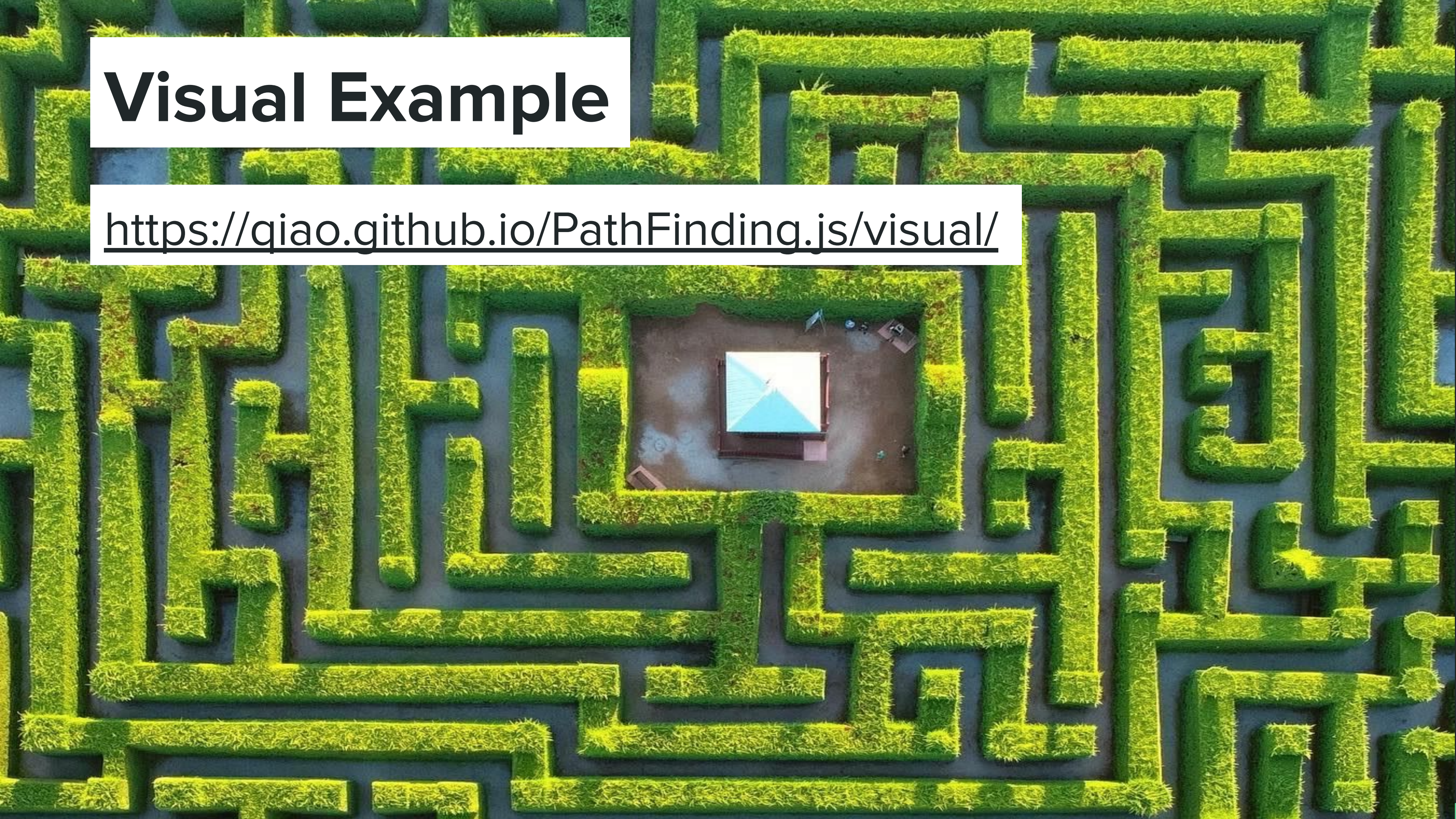
- Greedy Best-first Search
    - Similar to Breadth-first
    - Uses a heuristic h to estimate the distance from the goal
    - Sorts the queue/open list by the h value
    - Not optimal
- A* Search
    - Similar to the Greedy
    - Uses the total cost to reach the state as c
    - Sorts the queue/open list by the c+h value
    - Finds path with least cost

What is a Heuristic?

- Some form of ‘knowledge’ that can lead to a solution faster. Such as estimating how far away the goal is.

- It only finds an optimal solution if the heuristic is admissible (never overestimates).

- The different types of heuristics each have different ways of calculating their values.



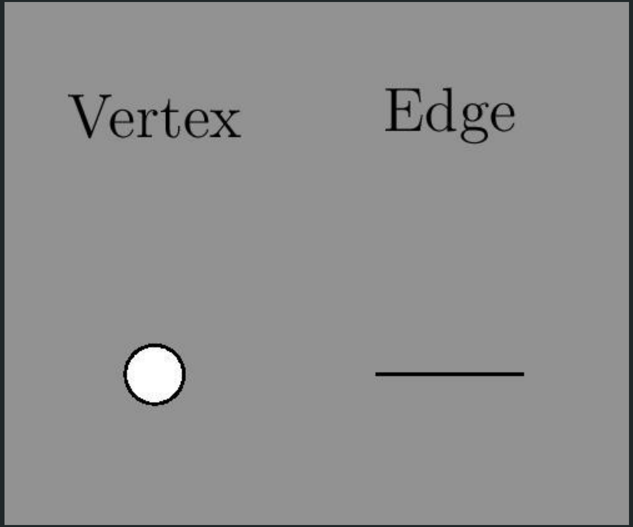
Visual Example

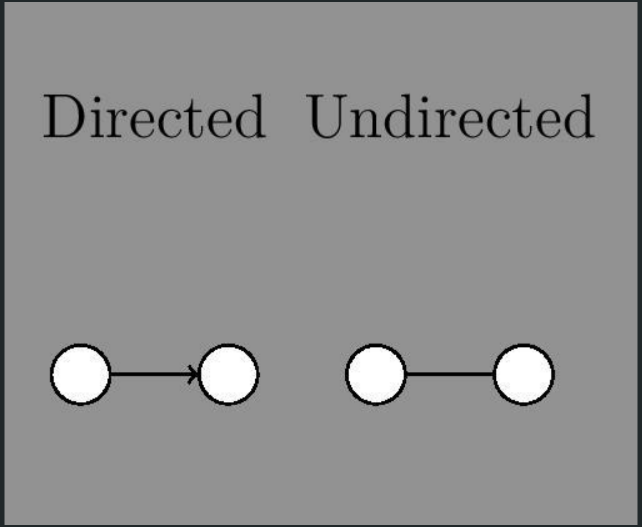
https://qiao.github.io/PathFinding.js/visual/

Search Algorithms

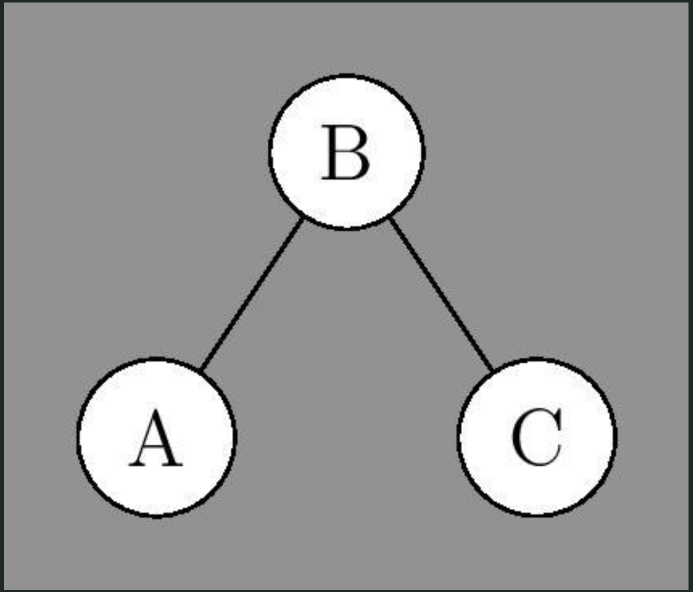
Rodney McCoy

Graph Theory - Review





Graph Theory - Review



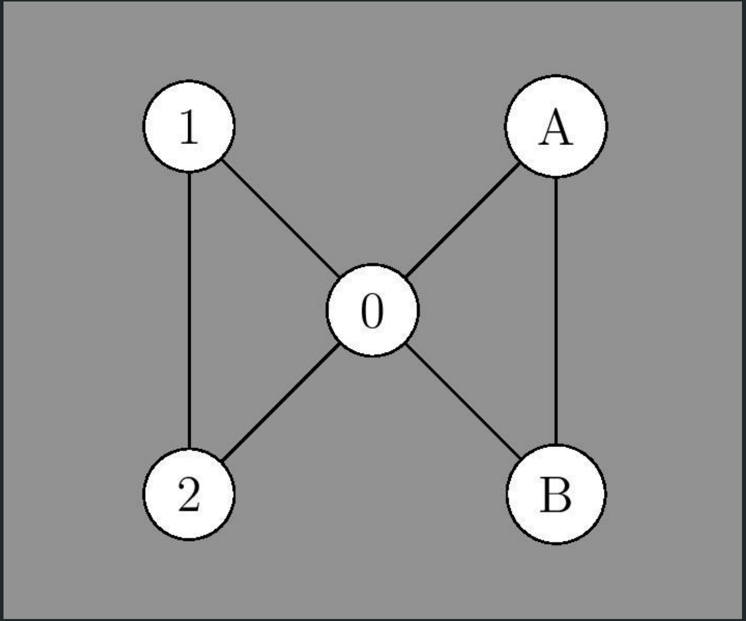
Two vertices are adjacent if

Their is a single edge separating them

The open neighborhood of a vertex v is

Every vertex that is adjacent to v

Graph Theory - Walk



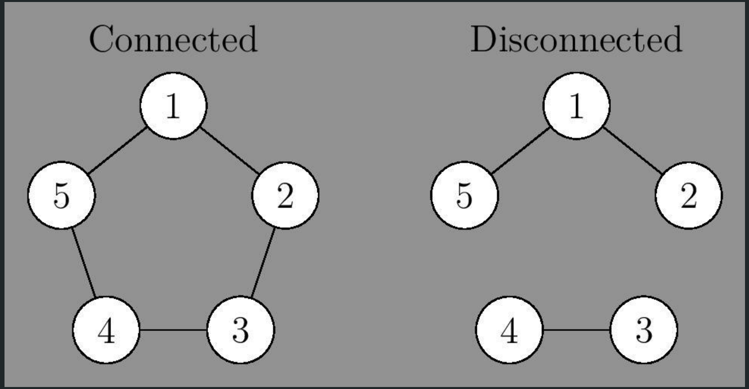
A walk is

A sequence of vertices such that the edges between each vertex is in the graph

A path is

A walk with distinct vertices (i.e. no repeated vertices)

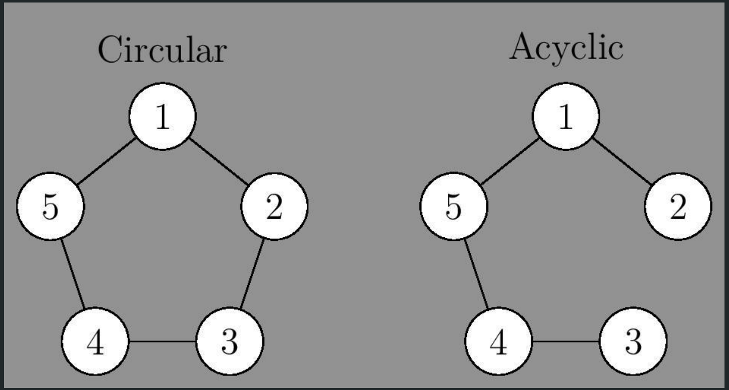
Graph Theory - Connected



A graph is connected if

Every pair of vertices can be joined by a path

Graph Theory - Cycles



A cycle is

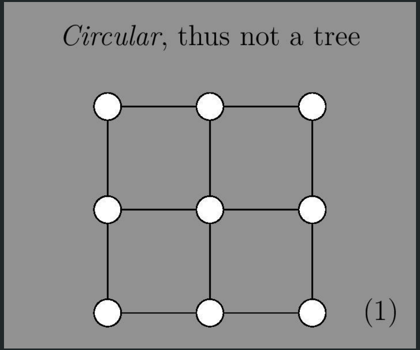
a path of length ≥ 3 with the first and last vertex being repeated

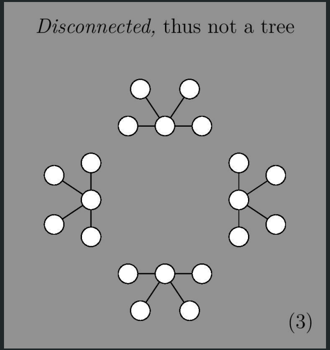
Ex: The 1-5 walk and (1, 5) edge represents the cycle of the vertices (1, 2, 3, 4, 5, 1)

Graph Theory - Trees

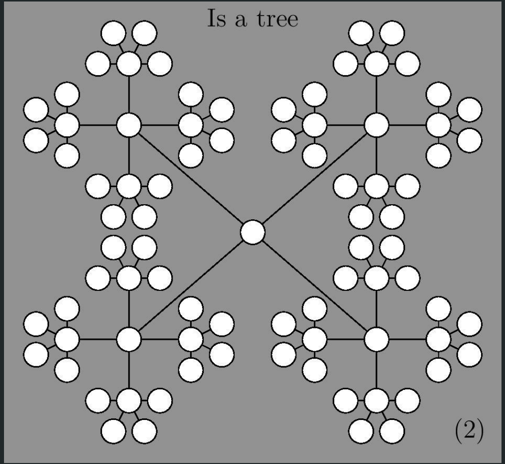
- A Tree is a Graph that is
  - Connected
  - Acyclic
  - **and undirected*

- I will use “Tree” and “Search Tree” interchangeably





This is called a forest



Graph Theory - Are Trees Directed or Undirected

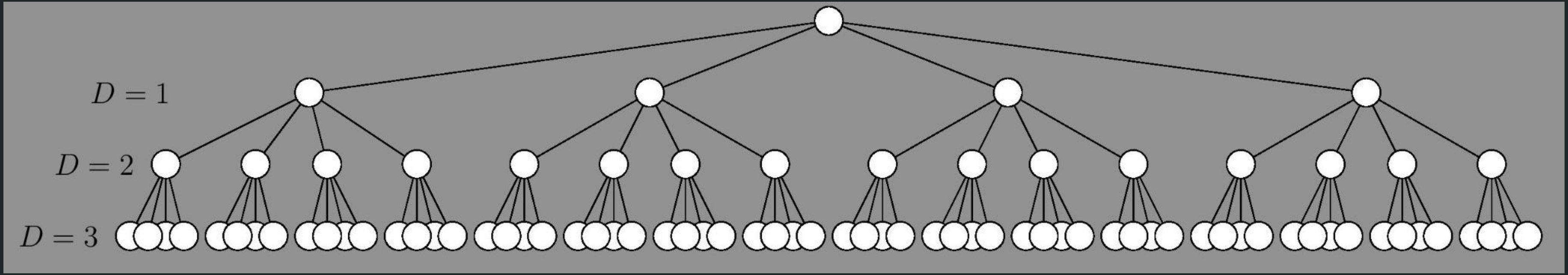
- In Mathematics
  - A Tree is undirected
- In Computer Science
  - A Tree is directed and rooted

- A Rooted Graph is
  - A graph where one node is labeled in a specific way to differentiate it from the rest

Graph Theory - Trees 2

- A Tree is a Graph that is
  - Connected
  - Acyclic

Example (2) from the previous



Branching Factor: 4

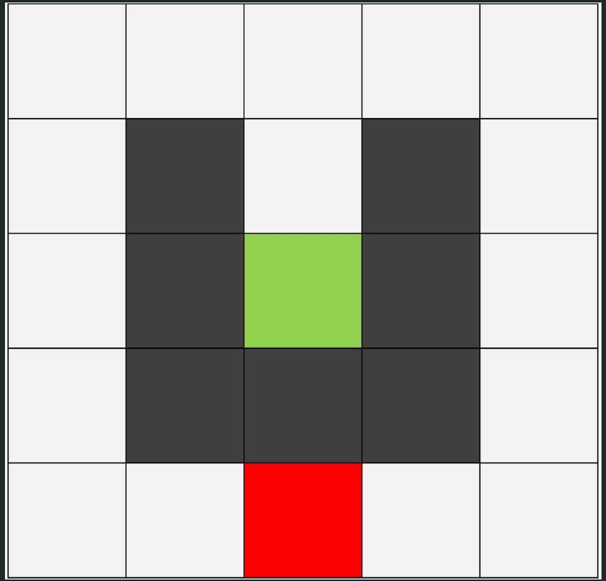
Depth: 3

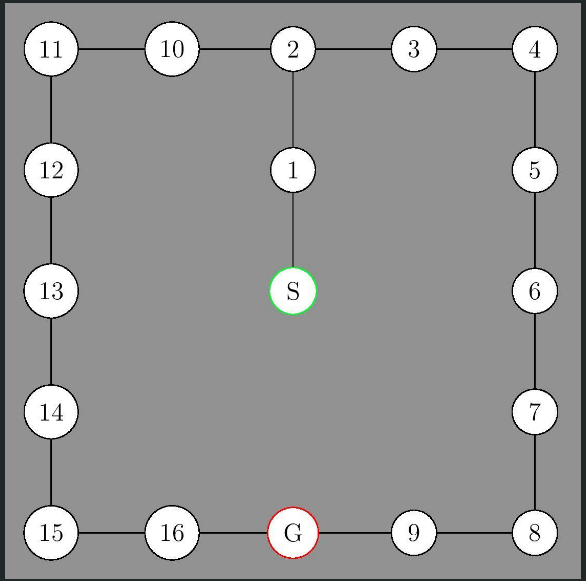
Search Algorithms

Pathfinding is a subset of the problems search algorithms can solve.

Any problem that can be represented with a search tree, can be solved using search algorithms.

Search Algorithms - Back to Pathfinding

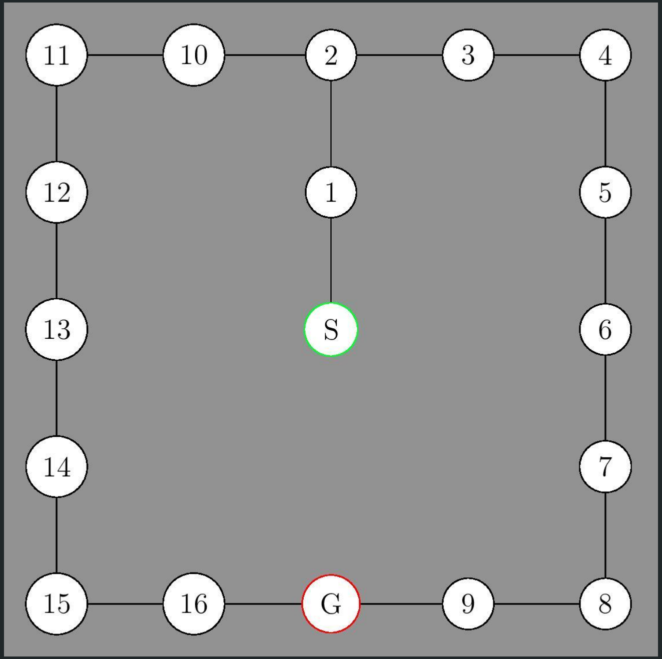




This pathfinding environment on the left can be represented as the graph on the right

Search Algorithms - Issues

- Clearly not a tree. Its circular.
- What happens when we perform a Greedy Best Search on this graph?

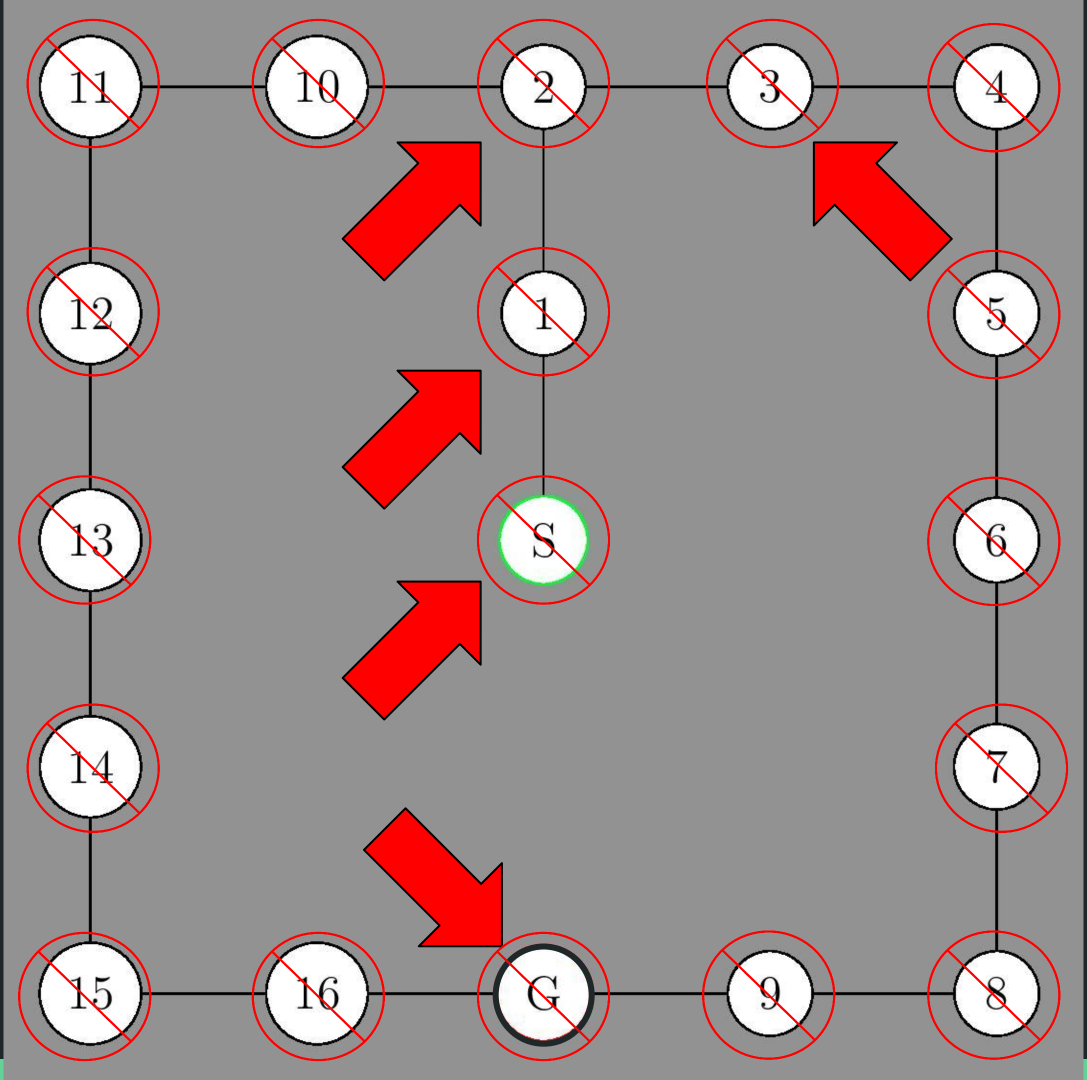


Search Algorithms

- If our search ignores vertices that are already discovered, we can ignore any cycles, which dynamically converts any graph into a tree
  - Therefore, any problem that can be represented with a graph, can be solved using search algorithms

DFS is complete.

So we know there is no solution



Any problem that can be represented with a search tree, or a graph, can be solved using search algorithms

So. What are some of these problems?

CSP

Constraint Satisfaction Problem

Formal Definition

3 Tuple (X, D, C)

- X is a set of variables
- D is the set of domains, one for each respective variable
- C is a set of constraints

CSP

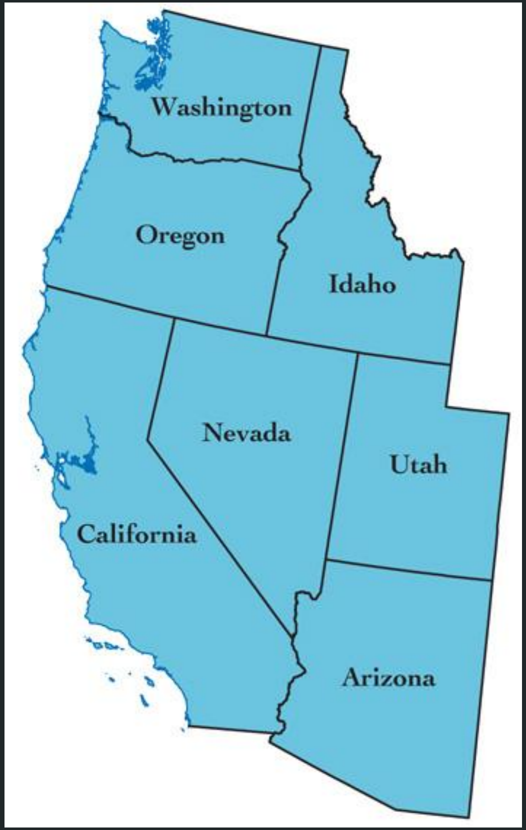
1. Define the constraints we want to satisfy

2. Apply a DFS to a search tree where we take each variable and assign it the next applicable value in its domain.

3. Backtrack if a constraint is violated *(Backtracking)*

4. Finish when every variable is assigned a value and no constraint is violated

Map Coloring Constraints

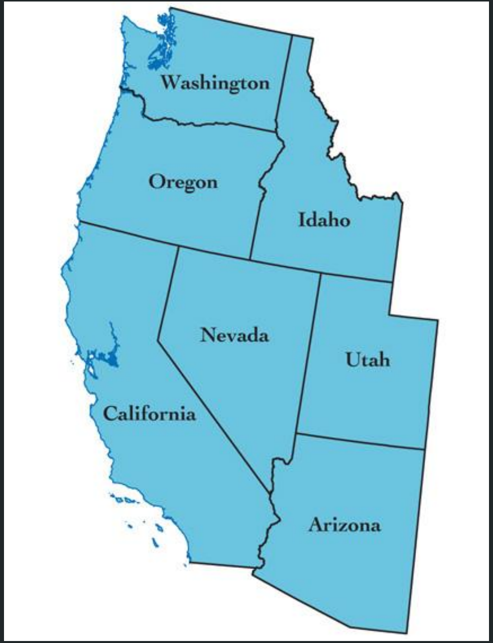


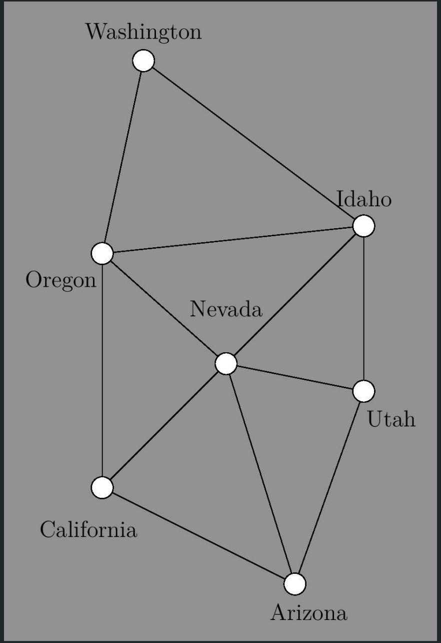
Create a graph to define our constraints where:

Each *vertex* is a region

Each *edge* is a border between two regions

Map Coloring Constraints





NOT THE SEARCH TREE.

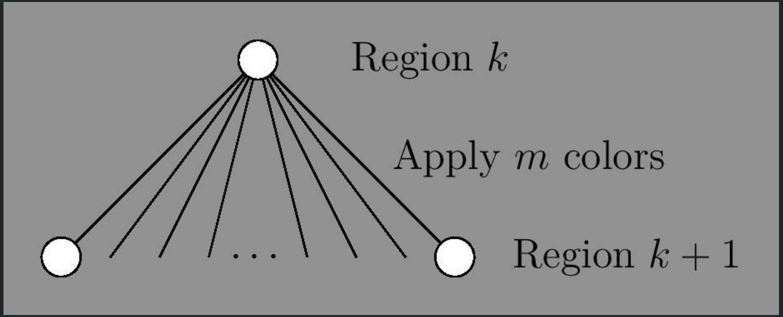
Just a representation of our problem as a graph to define the constraints we must satisfy, namely:

For every vertex, no vertex in its open neighborhood can have the same color as itself

Map Coloring Search Tree

- Each *vertex* is a region (variable)
- Each *edge* is a color applied to that region (value in domain)

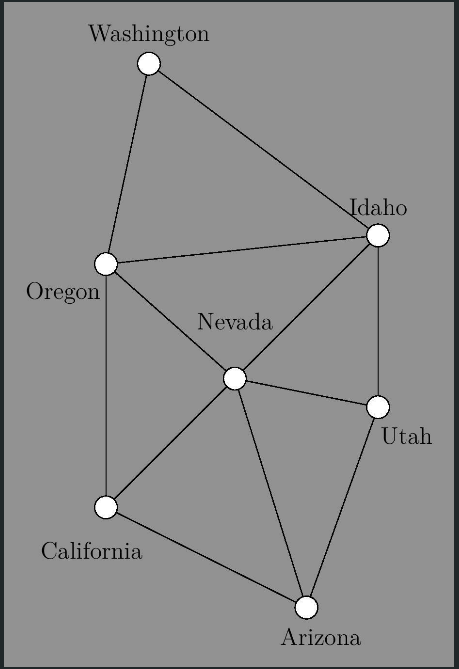
- Backtrack if a constraint is violated

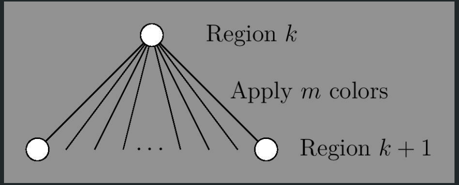


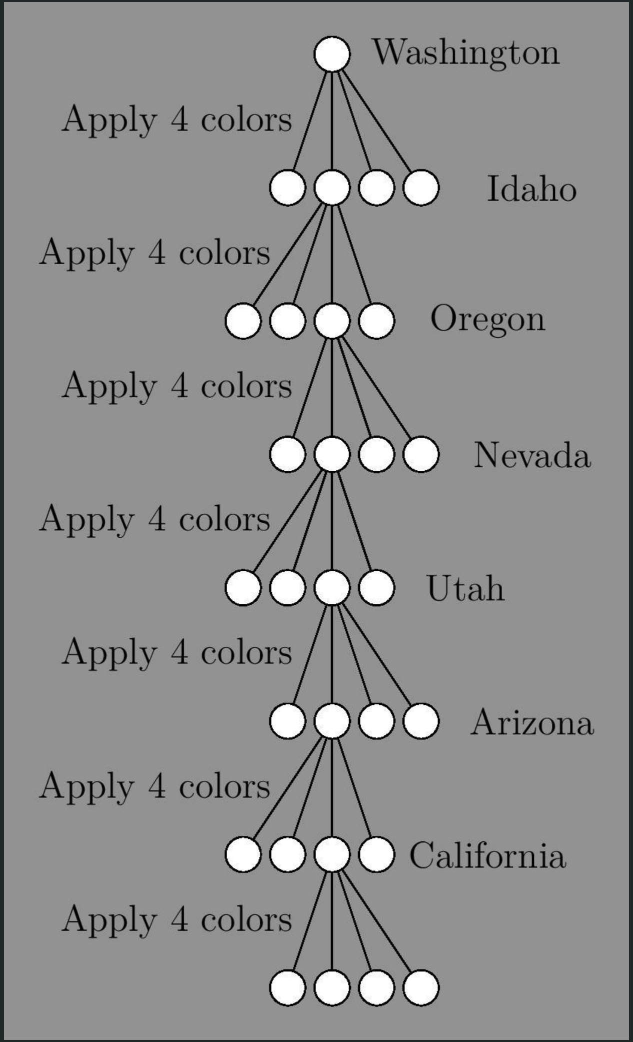
CSP - Map Coloring Search Tree

- Our search completes when we reach a leaf vertex without violating any constraints

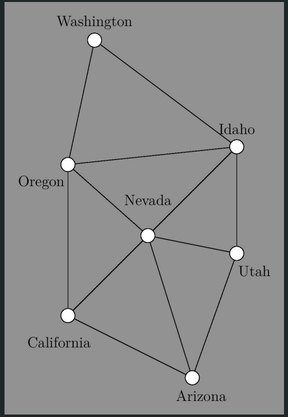
- A leaf is a
  - Vertex in a tree that is adjacent to only one other vertex



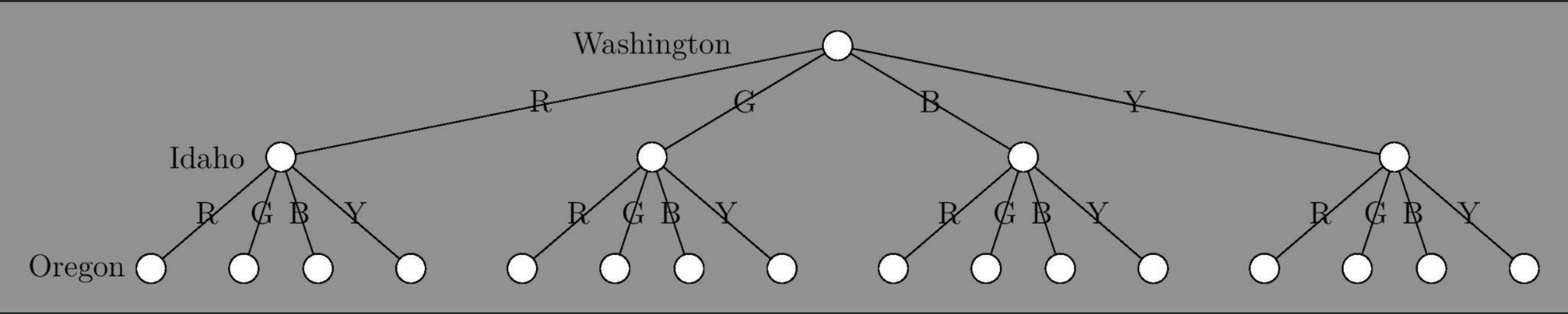




CSP - Map Coloring Problem



Search Tree



CSP - Summary

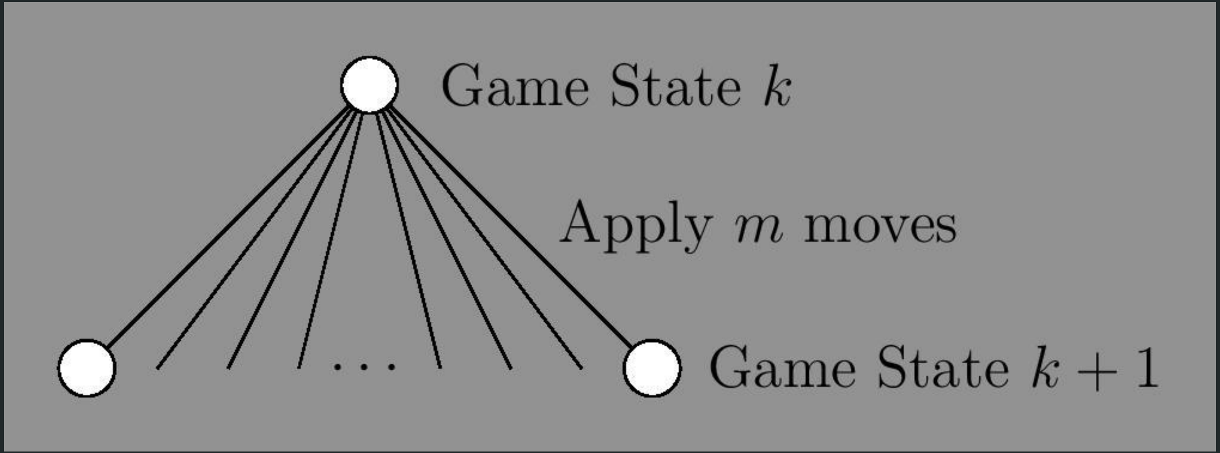
- We defined a graph to easily represent the constraints we had to satisfy
- We defined the search tree we wanted to DFS through. We then assigned a value to each variable, backtracked as soon as any constraint was violated, and completed once we reached a leaf vertex.

We don't explicitly store the discovered vertices since we are searching through a tree

Adversarial Search

We have a search tree.

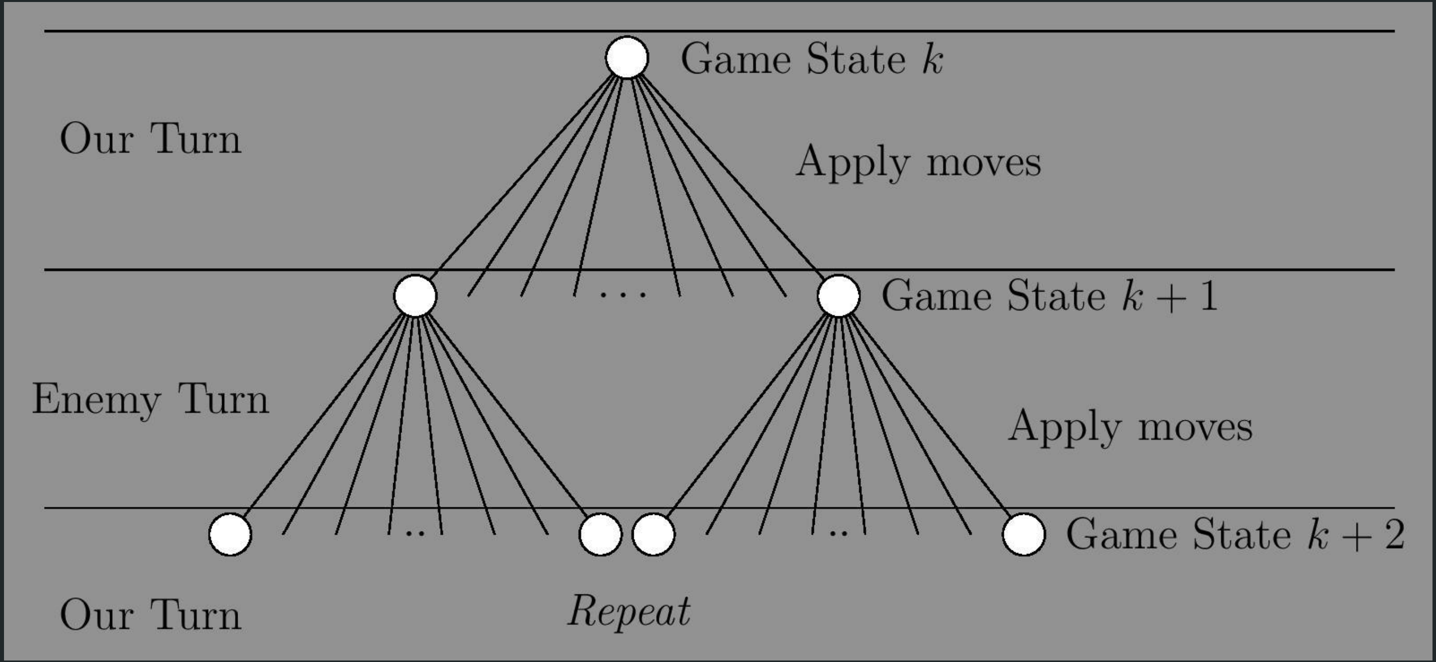
- Let each vertex be a game state.
- Let each edge be a game move.



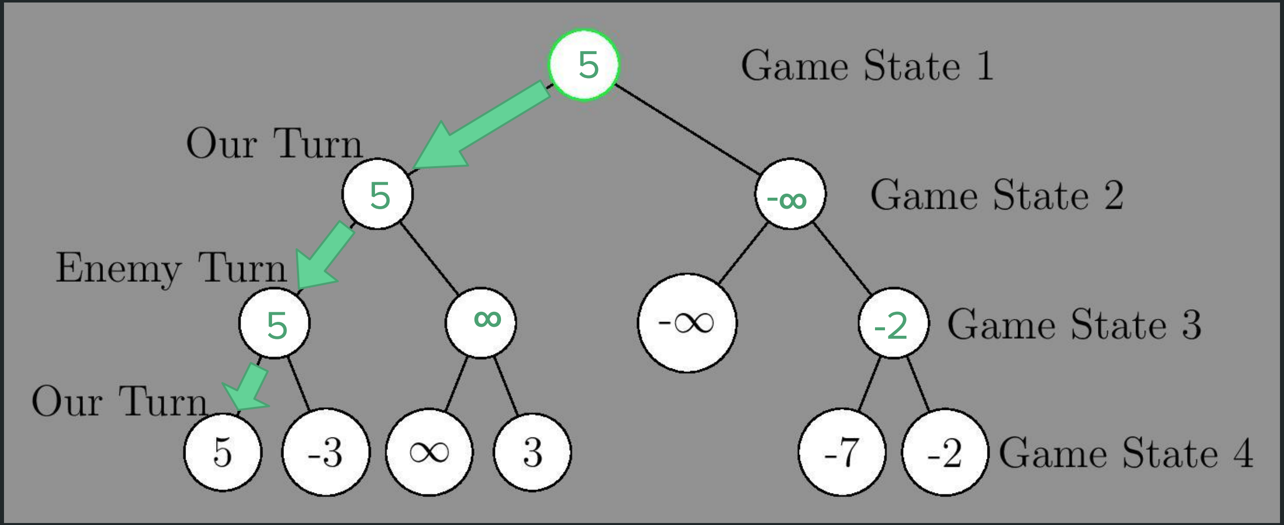
Adversarial Search

We have a search tree.

- Let each vertex be a game state.
- Let each edge be a game move.

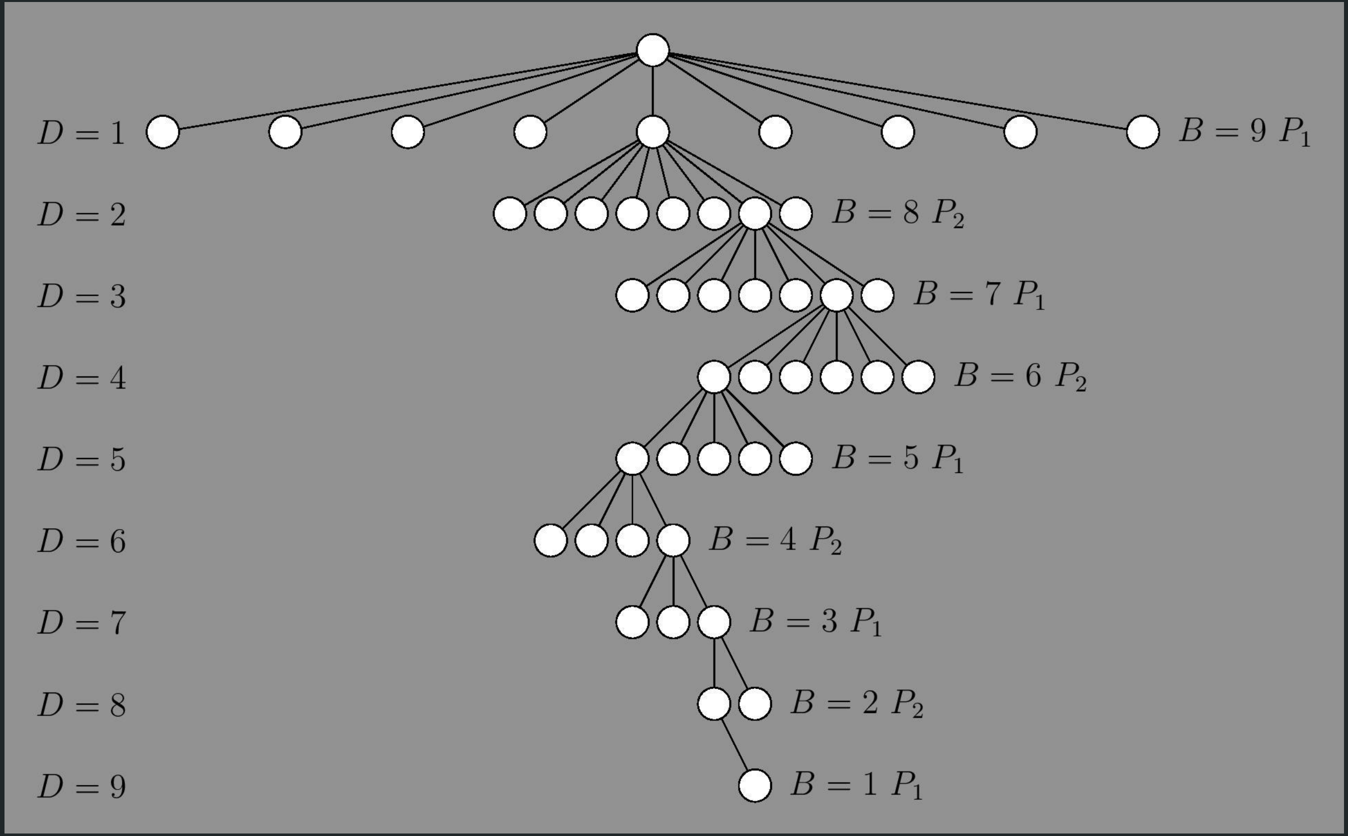


Adversarial Search



- Leaf vertices assigned utility by utility function
- Each non leaf vertex assigned value based off children (min-max)

Adversarial Search - Tic Tac Toe



- We backtrack when a game ending condition occurs (win / loss / draw).

Adversarial Search - Summary

- By performing an exhaustive DFS, we assign each leaf vertex a utility from the utility function, and assign each non-leaf vertex a utility dependent on its children plus which players turn it was.

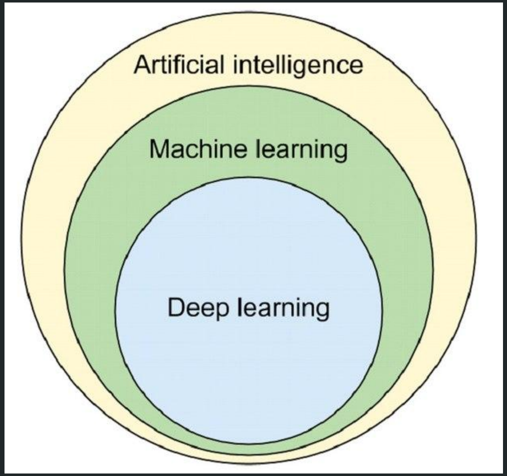
We don't explicitly store the discovered vertices since we are searching through a tree

Machine Learning/Neural Network

Trevor McGeary

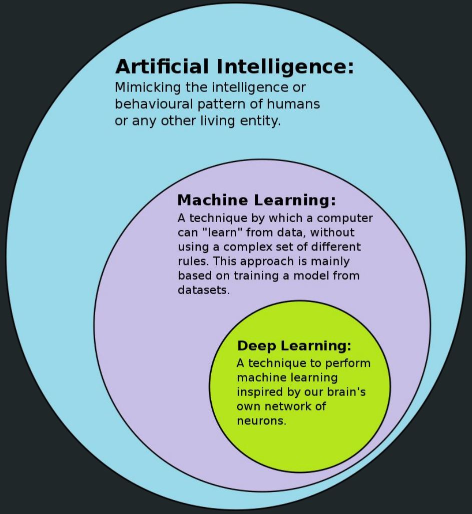
What is Machine Learning?

- Using algorithms and statistical models to make machines act without specific programming.
- Machine learning is a part of a greater whole in Artificial Intelligence
- Machine learning greatly involves the use of Deep Learning



Deep Learning

- Based on our brain's network of neurons
- Deep learning uses multiple layers of artificial neural networks to learn and solve tasks.
- Some applications of deep learning are image recognition, language processing, and advertising selection



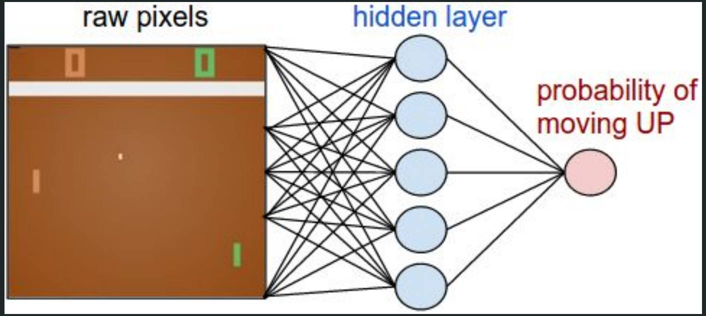
Artificial Intelligence:
Mimicking the intelligence or behavioural pattern of humans or any other living entity.
Machine Learning:
A technique by which a computer can "learn" from data, without using a complex set of different rules. This approach is mainly based on training a model from datasets.
Deep Learning:
A technique to perform machine learning inspired by our brain's own network of neurons.

Back to Machine Learning

- Machine Learning uses Intelligent Agents to solve problems
- Machine Learning is not widely used in video game development, but is used in playing video games.
- General Game Playing or Specific Game Playing
- Different methods of using machine learning to play video games, such as using deep learning agents or using computer vision.

Computer Vision Approach

- Utilizing the image recognition of deep learning, but taken to the next level
- Take input from screen data, pass through neural network, produce output corresponding to game input
- Examples: Pong (Pictured here), Doom (1993)



Intelligent Agent Approach

- Use intelligent agents to take the place of human players
- Seen in board games like Chess and Go
- In video games, seen in strategy games such as DOTA and Starcraft
- Agents have shown results to rival or even surpass human players in competition

Machine Learning to Play Games: Starcraft

- Starcraft is already a deeply complex game to play for humans
- Need to have knowledge of game units, game mechanics, and have knowledge of unit control.
- Also need to have a balance between big-picture economy management and low-level control of individual units

AlphaStar

- AlphaStar is generated by a deep learning neural network, that receives input from the game, and outputs instructions that constitute an action within the game.
- Additionally, AlphaStar's neural network was trained using replay data from players.
- Hundreds of intelligent agents were built that experienced the equivalent of 200 years of StarCraft playtime.
- Then, a competitive league was created were different agents played against each other, with strategies growing and evolving as the league went on.

AlphaStar

- After testing and training, AlphaStar was put to the test, and was able to not only reach the highest ranking level in Starcraft, but was able to defeat several professional players as well.
- AlphaStar was initially trained to only play one faction, against one faction, on one map.
- Later versions of AlphaStar would overcome these limitations.
- Due to the nature of StarCraft, DeepMind has speculated that the work done with AlphaStar could be applied to other things, such as weather prediction and climate modeling.

Further Resources

Fear AI Implementation

https://alumni.media.mit.edu/~jorkin/gdc2006_orkin_jeff_fear.pdf

Chess AI Implementation https://www.chessprogramming.org