



INDIAN INSTITUTE OF
INFORMATION
TECHNOLOGY

State Space Search

Dr. Animesh Chaturvedi

Assistant Professor: IIIT Dharwad

Young Researcher: Heidelberg Laureate Forum

Postdoc: King's College London & The Alan Turing Institute

PhD: IIT Indore MTech: IIITDM Jabalpur



Indian Institute of Technology Indore
भारतीय प्रौद्योगिकी संस्थान इंदौर



PDPM
Indian Institute of Information Technology,
Design and Manufacturing, Jabalpur

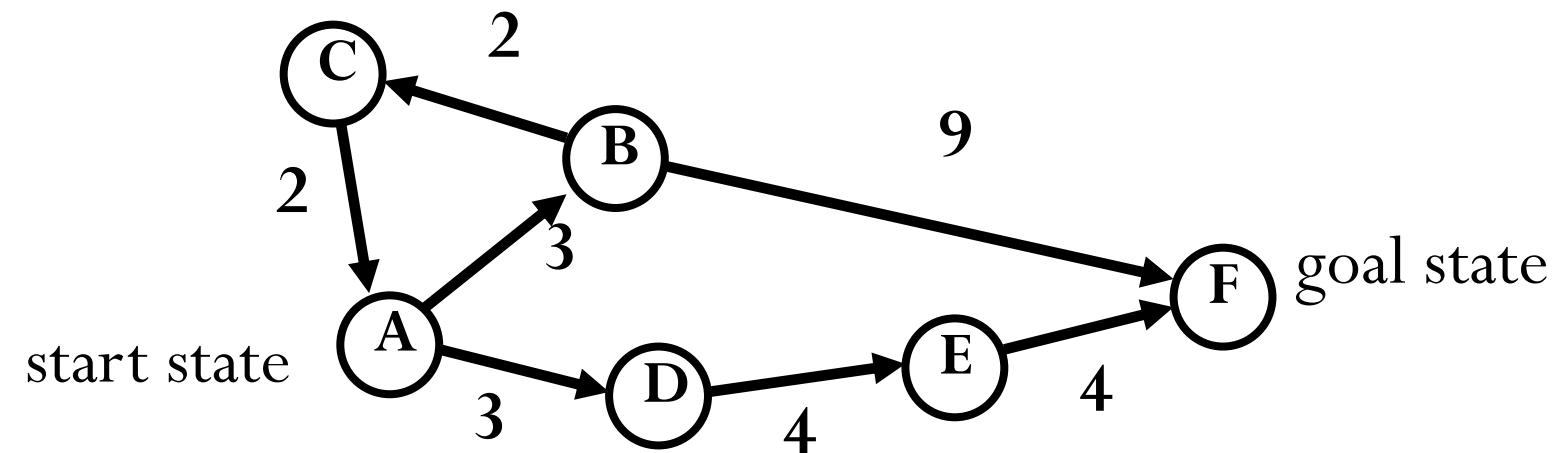
The
Alan Turing
Institute

Successive configurations or states are considered with intentions to find a goal state with desired properties.

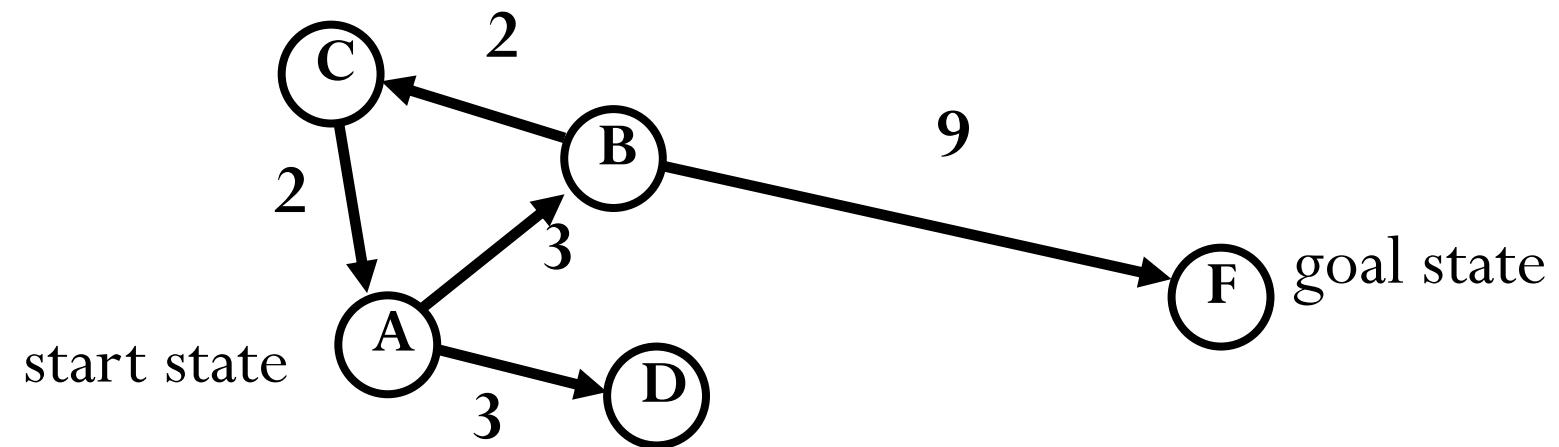
Search

- We have some actions that can change the state of the world
 - Change induced by an action perfectly predictable
- Try to come up with a sequence of actions that will lead us to a goal state
 - May want to minimize number of actions
 - More generally, may want to minimize total cost of actions
- Do not need to execute actions in real life while searching for solution!
 - Everything perfectly predictable anyway

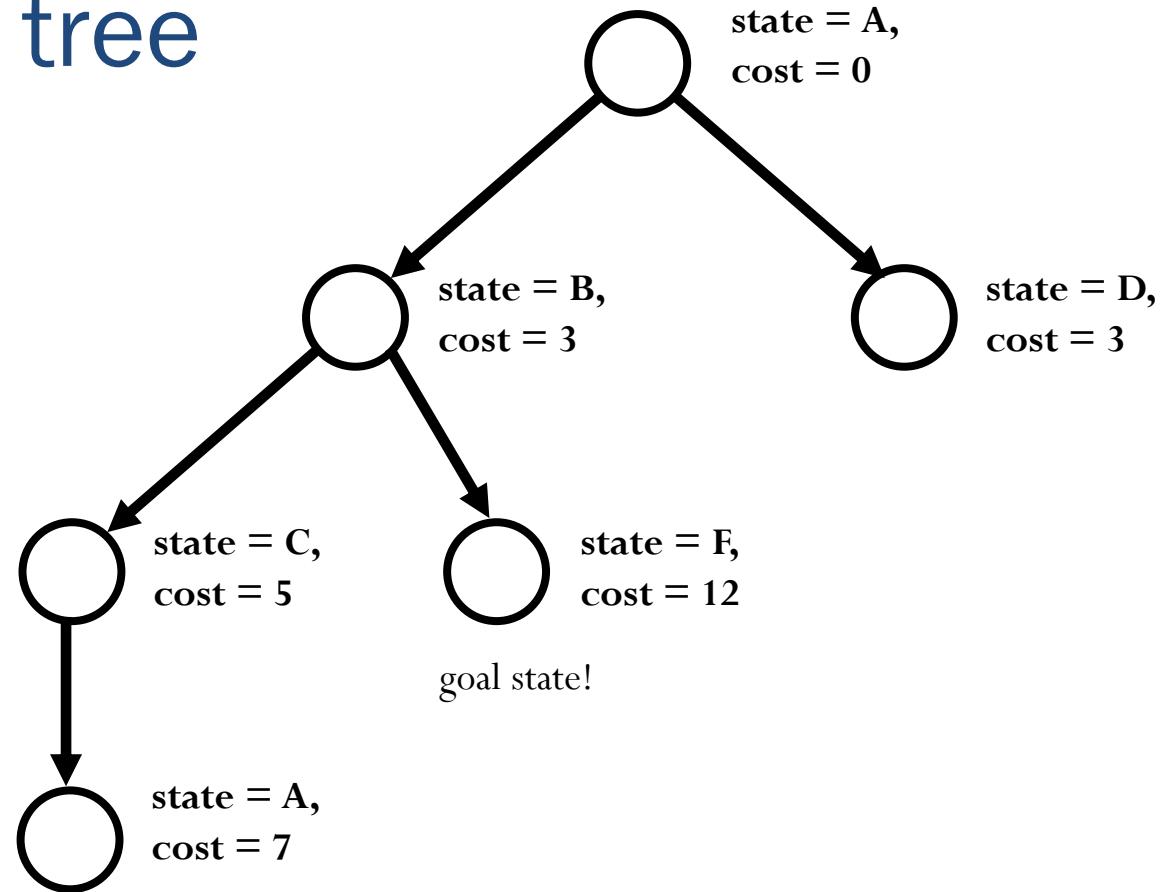
A simple example: traveling on a graph



Searching for a solution

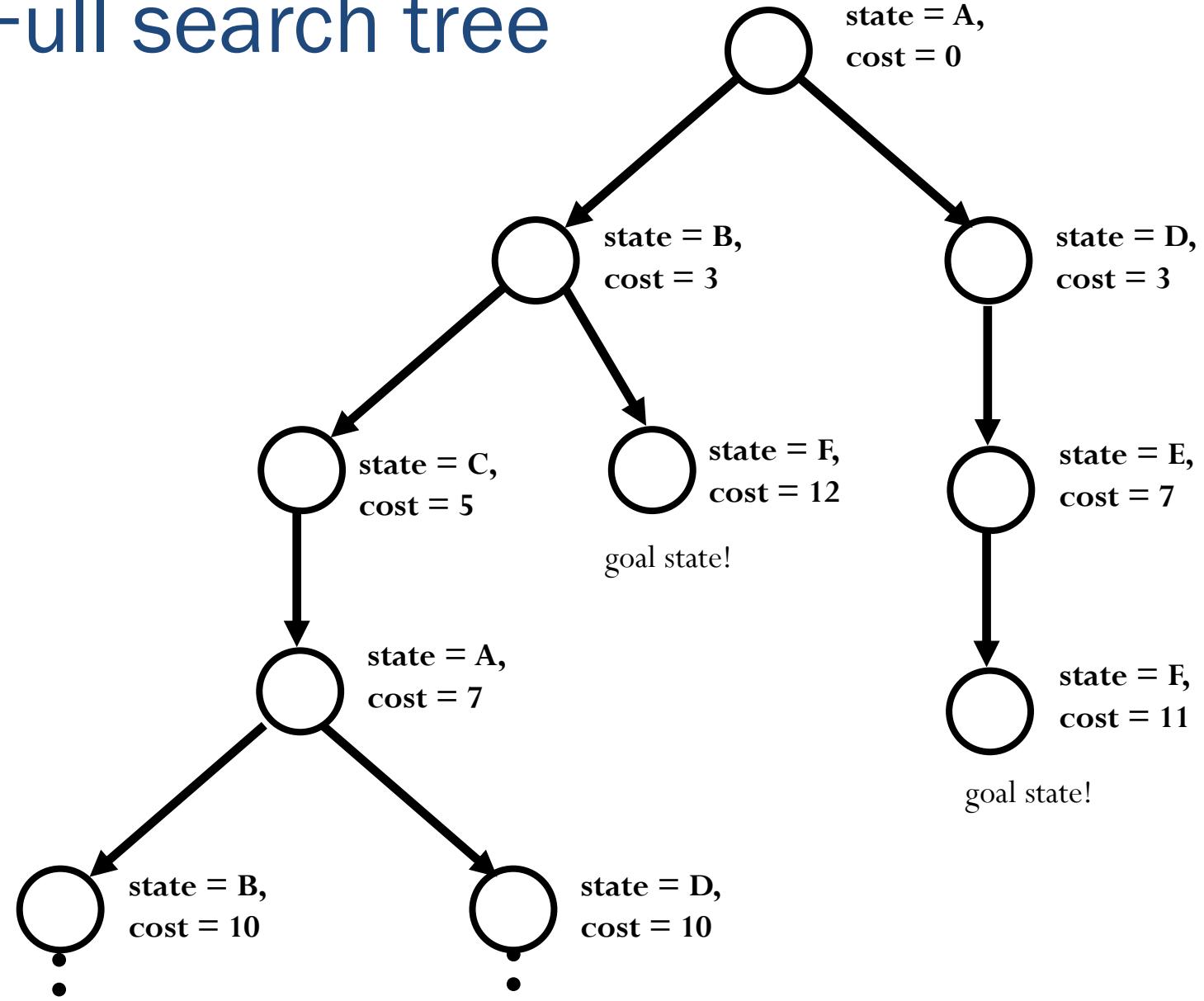


Search tree

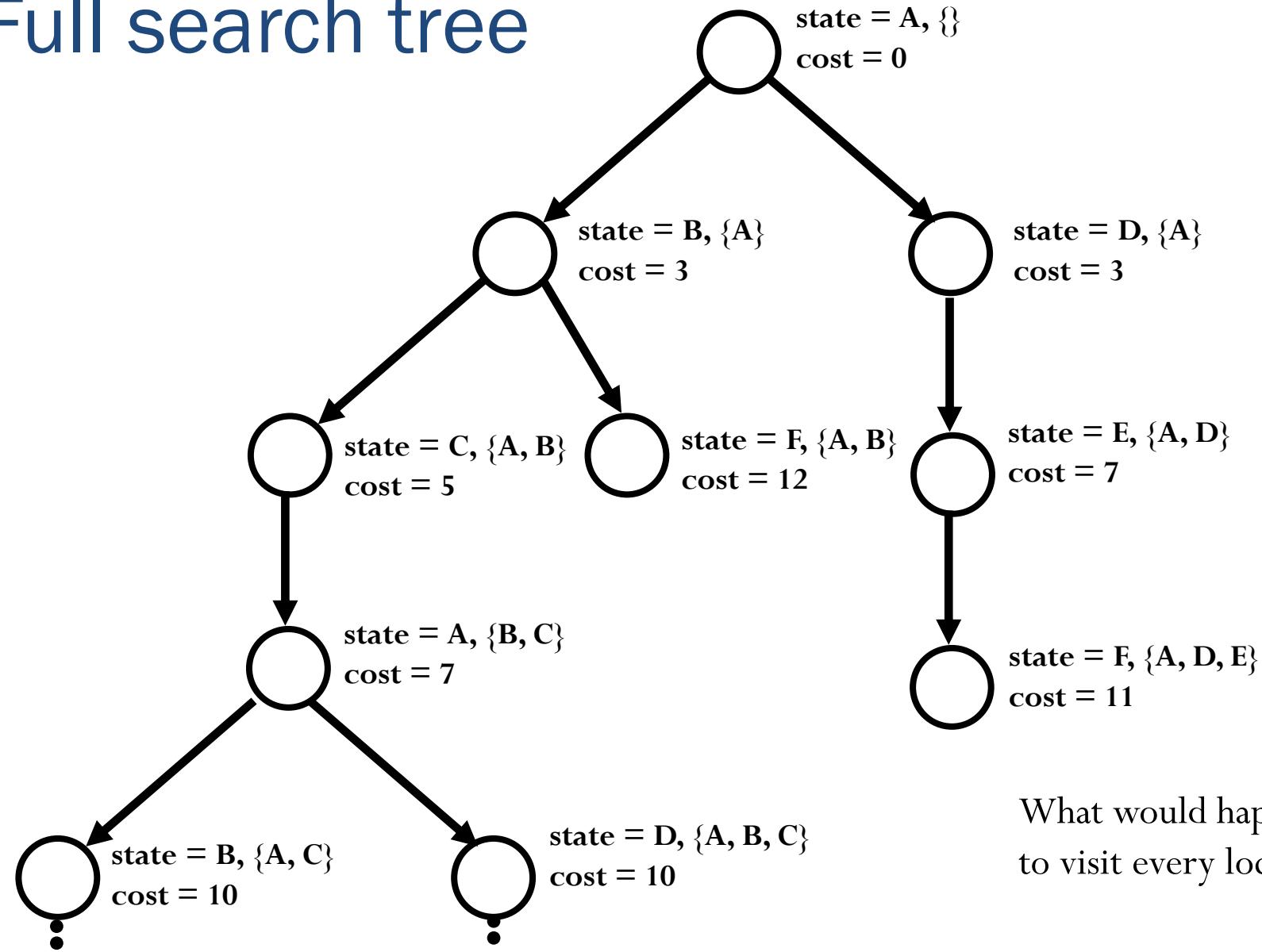


search tree nodes and states are not the same thing!

Full search tree



Full search tree



What would happen if the goal were to visit every location twice?

Key concepts in search

- Set of **states** that we can be in
 - Including an **initial state**...
 - ... and **goal states** (equivalently, a **goal test**)
- For every state, a set of **actions** that we can take
 - Each action results in a new state
 - Typically defined by **successor function**
 - Given a state, produces all states that can be reached from it
- **Cost function** that **determines the cost of each action** (or **path** = sequence of actions)
- **Solution:** path from initial state to a **goal state**
 - **Optimal solution:** solution with minimal cost

Brute-force way of searching

Uninformed search

Various types of uninformed searches are

- * BFS
- * DFS
- * Depth-limiting Search
- * Iterative Deepening DFS
- * Uniform Cost Search
- * Bidirectional Search

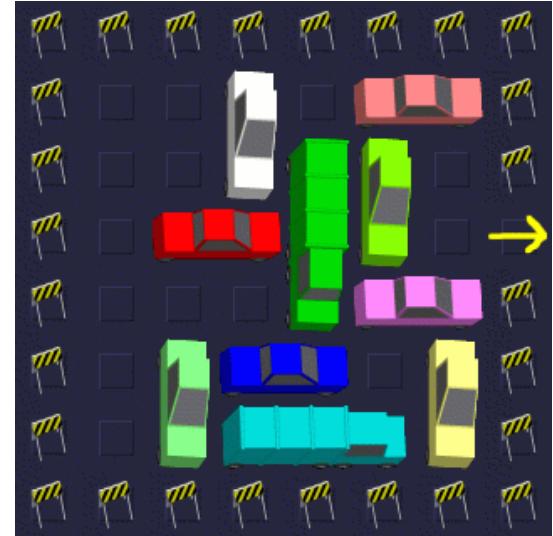
Uninformed search

- Given a state, we only know whether it is a goal state or not
- Cannot say one nongoal state looks better than another nongoal state
- Can only traverse state space blindly in hope of somehow hitting a goal state at some point
 - Also called **blind search**
 - Blind does **not** imply unsystematic!

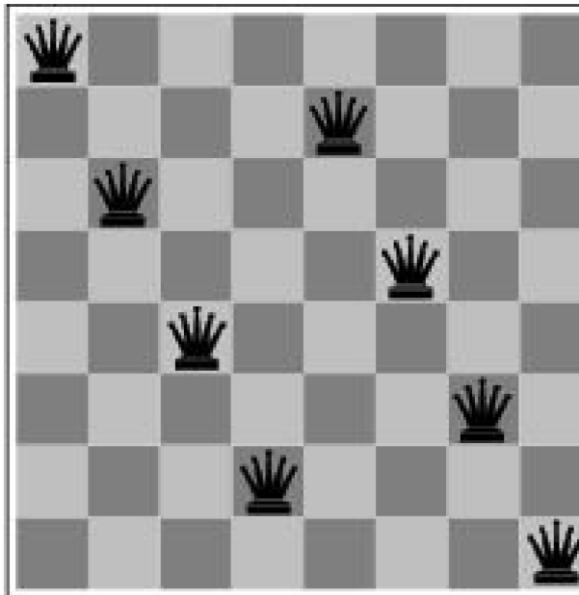
Searching Examples



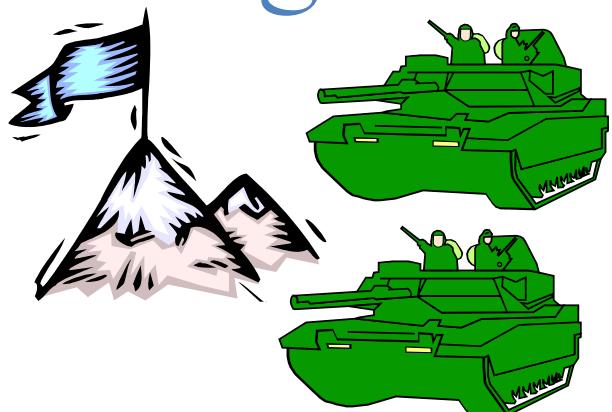
Rush Hour: Move cars forward
and backward to “escape”



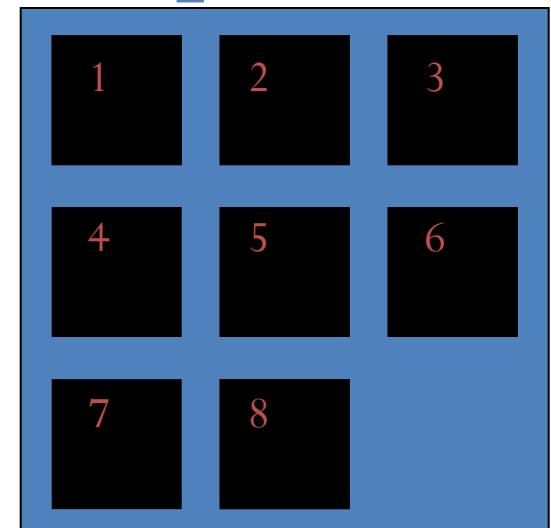
8-queens problem



Logistics



8-puzzle



In a genetic algorithm, a population of candidate solutions to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties which can be mutated and altered.

The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected

from the current population, and each individual's properties are modified to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

Generic search algorithm

- Fringe = set of nodes generated but not expanded
- fringe := {initial state}
- loop:
 - if fringe empty, declare failure
 - choose and remove a node v from fringe
 - check if v's state s is a goal state; if so, declare success
 - if not, expand v, insert resulting nodes into fringe
- Key question in search: Which of the generated nodes do we expand next?

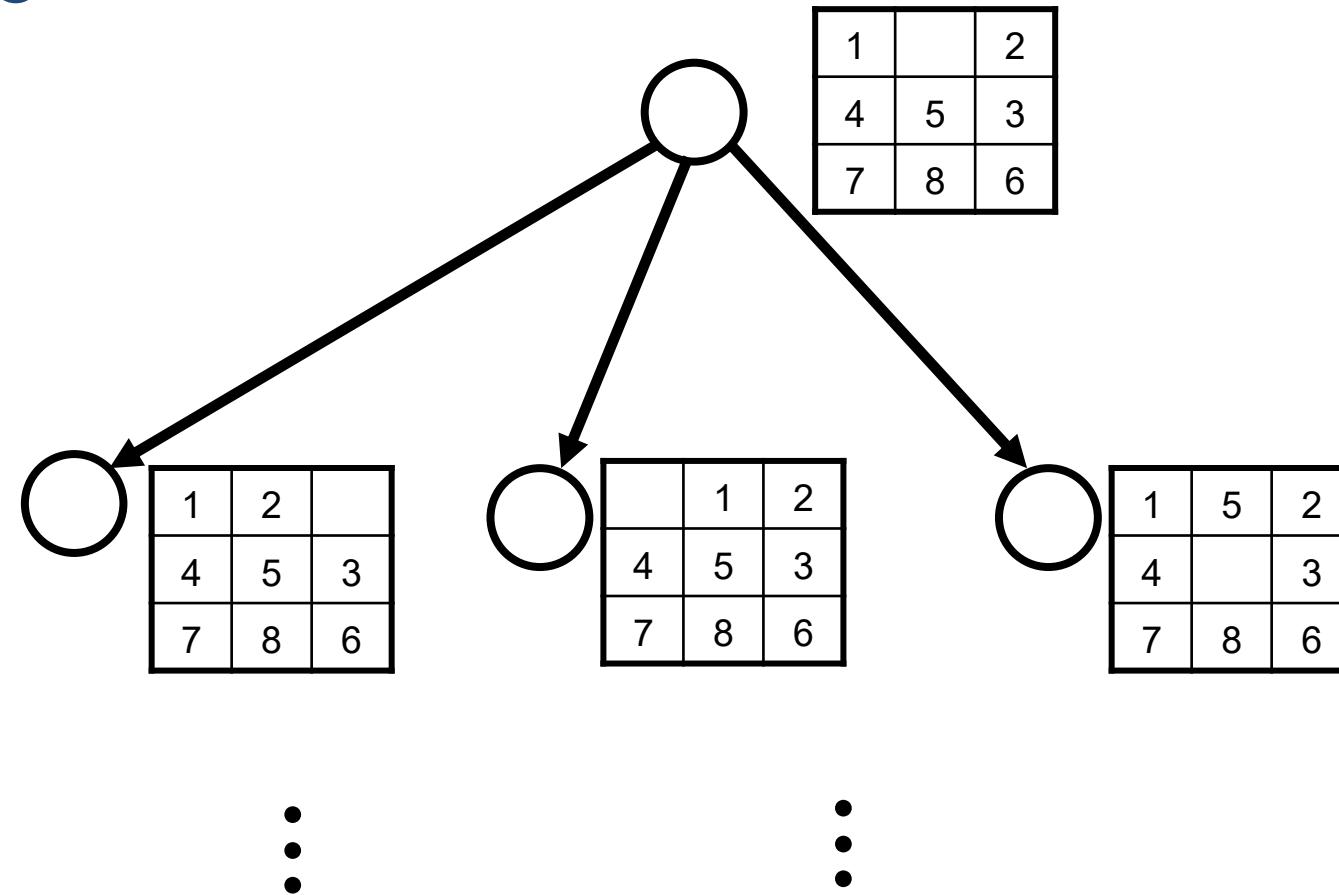
8-puzzle

| | | |
|---|---|---|
| 1 | | 2 |
| 4 | 5 | 3 |
| 7 | 8 | 6 |

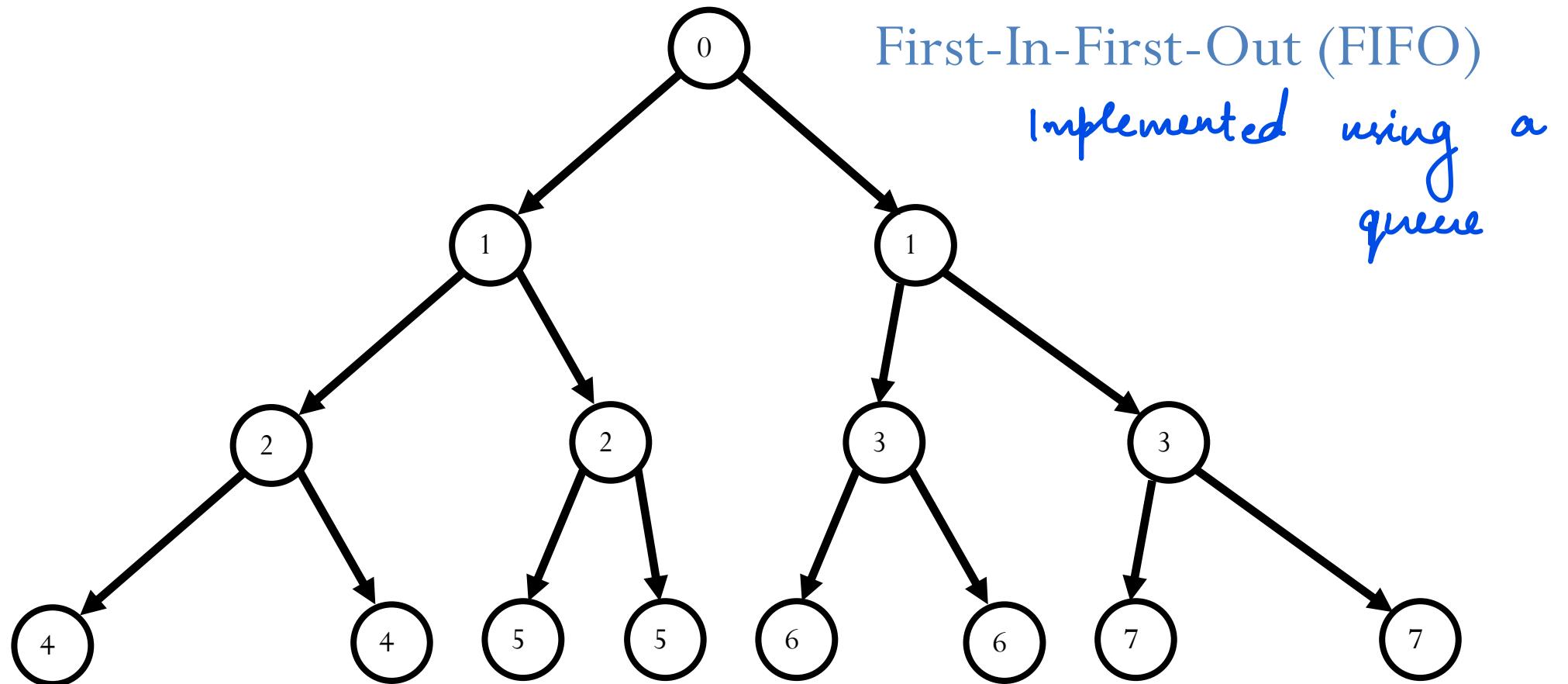
| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

goal state

8-puzzle



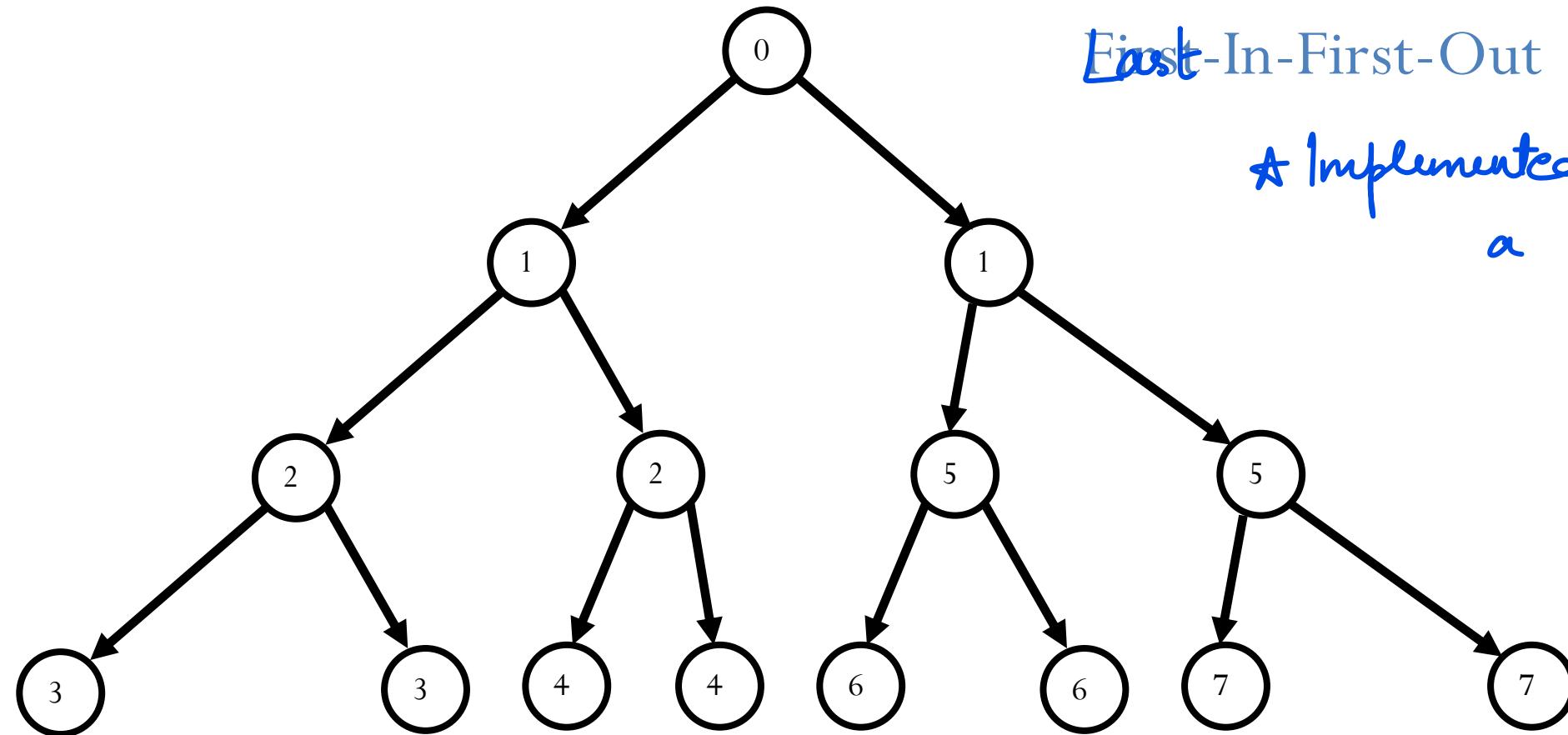
Breadth-First Search



Properties of Breadth-First Search

- Nodes are expanded in the same order in which they are generated
 - Fringe can be maintained as a First-In-First-Out (FIFO) queue
- BFS is **complete**: if a solution exists, one will be found
- BFS finds a **shallowest** solution
 - Not necessarily an optimal solution
- If every node has b successors (the **branching factor**), first solution is at depth d , then fringe size will be at least b^d at some point
 - This much space (and time) required ☹

Depth-First Search



First-In-First-Out

* Implemented using
a stack

Implementing Depth-First Search

- Fringe can be maintained as a Last-In-First-Out (LIFO) queue (aka. a stack)
- Also easy to implement recursively:
 - $\text{DFS}(\text{node})$
 - If $\text{goal}(\text{node})$ return $\text{solution}(\text{node})$;
 - For each successor of node
 - Return $\text{DFS}(\text{successor})$ unless it is *failure*;
 - Return *failure*;

Properties of depth-first search

- Not complete (might cycle through nongoal states)
- If solution found, generally not optimal/shallowest
- If every node has b successors (the **branching factor**), and we search to at most depth m , fringe is at most b^m
 - Much better space requirement ☺
 - Actually, generally don't even need to store all of fringe
- Time: still need to look at every node
 - $b^m + b^{m-1} + \dots + 1$ (for $b > 1$, $O(b^m)$)
 - **Inevitable** for uninformed search methods...

Combining good properties of BFS and DFS

- Limited depth DFS: just like DFS, except never go deeper than some depth d
 - Iterative deepening DFS:
 - Call limited depth DFS with depth 0;
 - If unsuccessful, call with depth 1;
 - If unsuccessful, call with depth 2;
 - Etc.
 - Complete, finds shallowest solution
 - Space requirements of DFS
 - May seem wasteful timewise because replicating effort
 - Really not that wasteful because **almost all effort at deepest level**
 - $db + (d-1)b^2 + (d-2)b^3 + \dots + 1b^d$ is $O(b^d)$ for $b > 1$
- can fail in 2 scenarios*
- Standard failure value - solution to the problem doesn't exist*
- cutoff failure value - solution doesn't exist within the defined depth limit*

Searching solution evaluation

- Comparing multiple searching algorithm based on
 - Completeness: does it always find a solution if one exist?
 - Time complexity: How long depends on number of nodes
 - Space complexity: Memory depends on number of nodes
 - Optimality: Find shortest path (or least cost solution)?
 - Systematicity: does it visit each state at most once?

Iterative Deepening DFS - This algorithm performs DFS upto depth-limit & keeps increasing depth-limit until goal state is found.

Combines fast search of BFS and memory efficiency of DFS

Useful when search space is large & depth of goal state is unknown.

Depth vs. Breadth-first

Let $|T(s)| \leq b$ (branching factor), goal at depth d

- How implement priority queue?
- Completeness?
- Time complexity?
- Space complexity?
- Optimality?



Breath First Search

- Completeness?
 - Yes
- Time complexity?
 - $O(b^d)$
- Space complexity?
 - $O(b^d)$ 😞
- Optimality?
 - yes

Depth First Search

- Completeness?
 - Yes, assuming state space finite
- Time complexity?
 - $O(|V+E|)$, can do well if lots of goals
- Space complexity?
 - $O(n)$, n deepest point of search
- Optimality?
 - No 😞

Let b as branching factor with goal at depth d

~~Imp~~

Depth-limited Search

DFS, only expand nodes depth $\leq l$.

- Completeness?
 - No, if $l \leq d$. ☹
- Time complexity?
 - $O(b^l)$
- Space complexity?
 - $O(l)$
- Optimality?
 - No ☹

Iterative Deepening

Depth limited, increasing l .

- Completeness?
 - Yes. ☺
- Time complexity?
 - $O(b^d)$, even with repeated work! ☺
- Space complexity?
 - $O(d)$ ☺
- Optimality?
 - Yes ☺

Bidirectional search

- Even better: search from both the start and the goal, in parallel!
- If the shallowest solution has depth d and branching factor is b on both sides, requires only $O(b^{d/2})$ nodes to be explored!

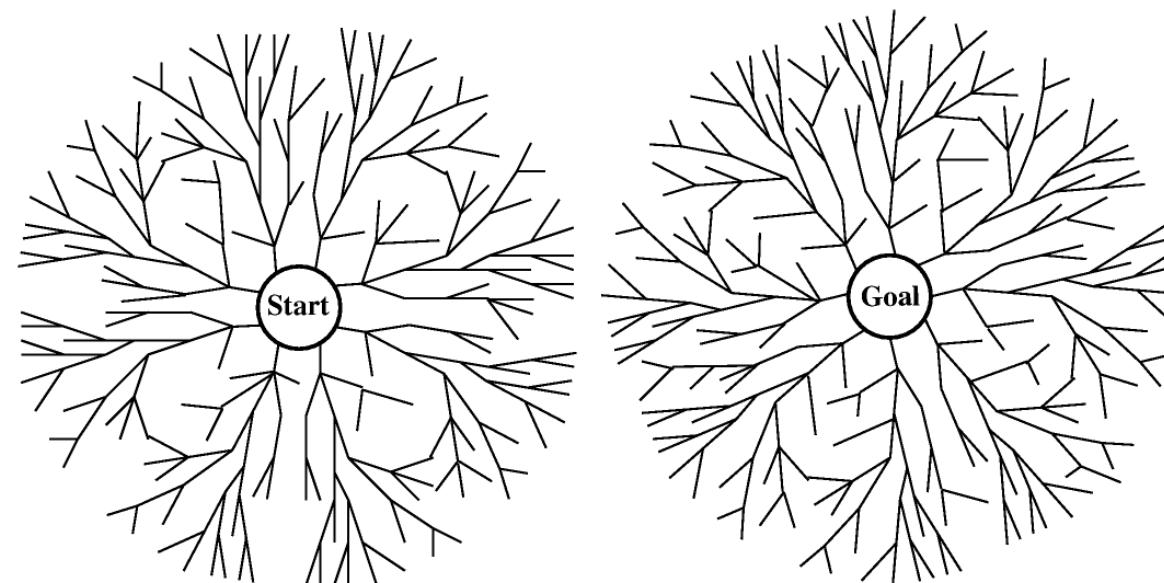


image from cs-alb-pc3.massey.ac.nz/notes/59302/fig03.17.gif

Bidirectional Search

BFS in both directions

Need N^{-1}

How could this help?

- b^1 vs $2b^{1/2}$

What makes this hard to implement?

Runs 2 simultaneous searches

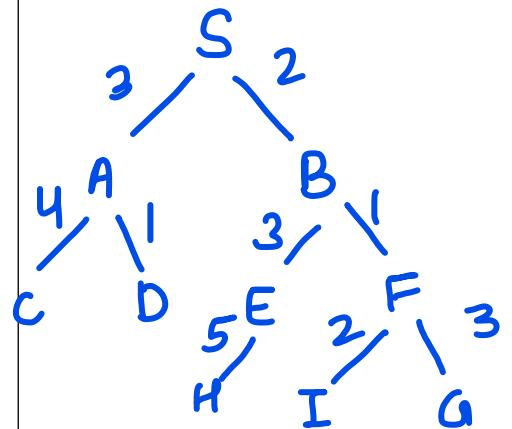
- ① FORWARD SEARCH - from initial state to goal state
- ② BACKWARD SEARCH - from goal state to initial state

ends when 2 searchers intersect

- * Fast & requires less memory
- * Goal state should be known in advance, Difficult to implement for trees

Informed Search

Aus



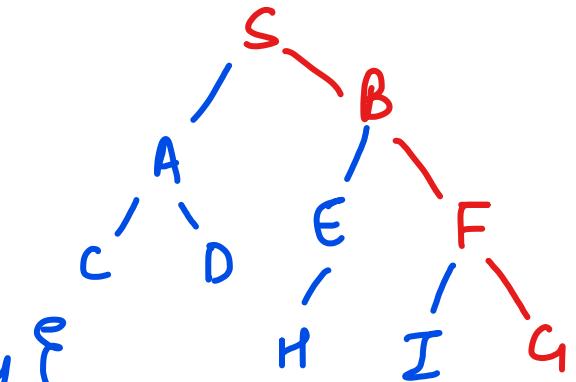
① open = { A, B } closed = { S }

② open = { E, F, A } closed = { S, B }

③ open = { I, G, E, A } closed = { S, B, F }

④ open = { G, E, A } closed = { S, B, F, G }

| node | S | A | B | C | D | E | F | G | H | I |
|--------|----|----|---|---|---|---|---|---|---|---|
| $H(n)$ | 13 | 12 | 4 | 7 | 3 | 8 | 2 | 0 | 4 | 9 |



Informed search

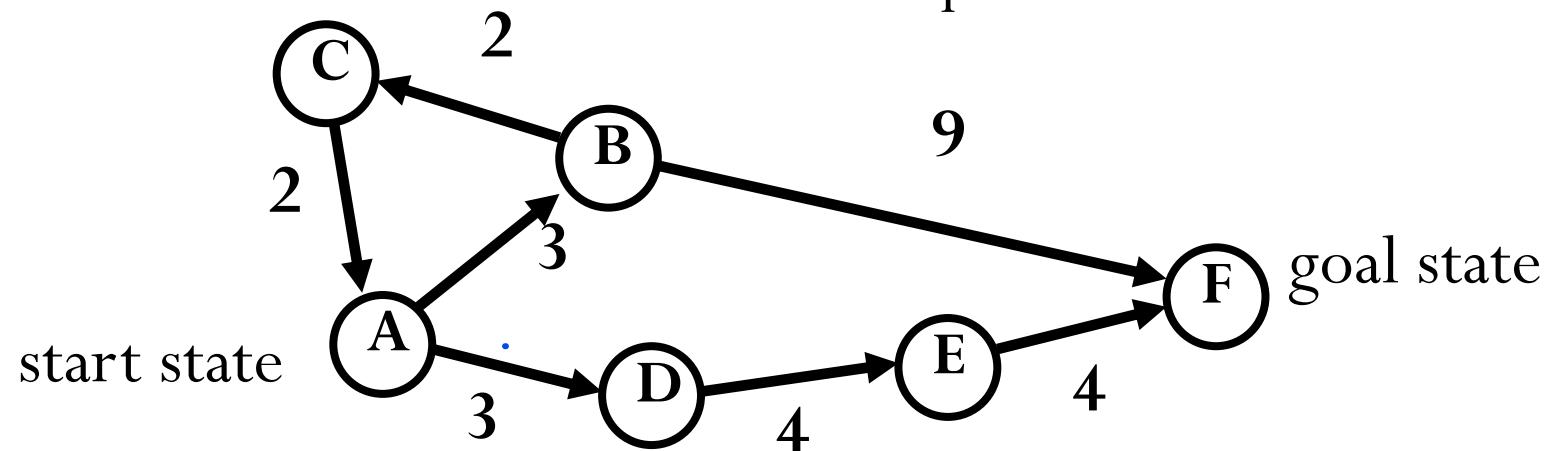
- So far, have assumed that **no nongoal state looks better than another**
- Unrealistic
 - Even without knowing the road structure, some locations seem closer to the goal than others
 - Some states of the 8s puzzle seem closer to the goal than others
- Makes sense to expand closer-seeming nodes first

Heuristic function takes current state as the input and outputs an approximation of how close it is to goal state.

Heuristic doesn't always provide best solution, but gives a good solution in less time. Heuristic function is always positive!

Pure heuristic - CLOSED (nodes already expanded upon)
Heuristics OPEN (nodes that can be still expanded)
We expand the OPEN node with lowest heuristic.

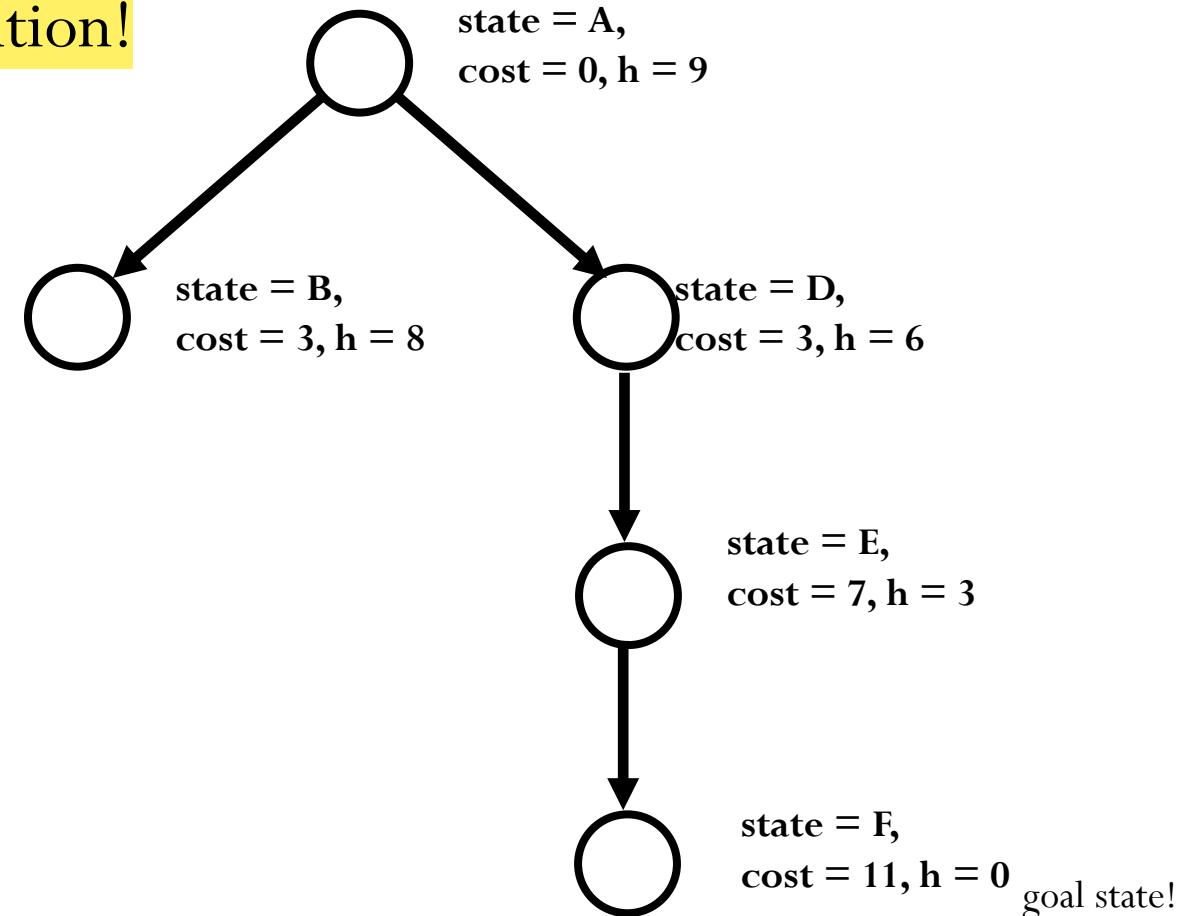
- Key notion: heuristic function $h(n)$ gives an estimate of the distance from n to the goal
 - $h(n)=0$ for goal nodes
- E.g. straight-line distance for traveling problem
- Say: $h(A) = 9, h(B) = 8, h(C) = 9, h(D) = 6, h(E) = 3, h(F) = 0$
- We're adding something new to the problem!
- Can use heuristic to decide which nodes to expand first



Greedy best-first search

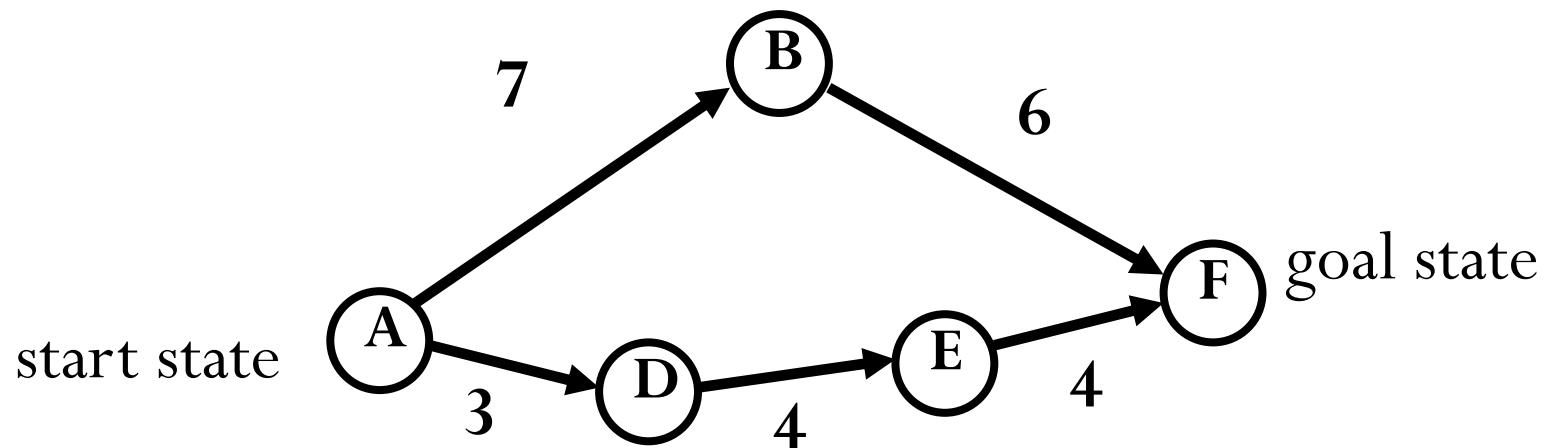
- Greedy best-first search: expand nodes with lowest h values first
- Rapidly finds the optimal solution!
- Does it always?

- Not optimal
- Gets stuck in loops



A bad example for greedy

- Say: $h(A) = 9, h(B) = 5, h(D) = 6, h(E) = 3, h(F) = 0$
- Problem: greedy evaluates the promise of a node only by how far is left to go, does not take cost occurred already into account



IDA*

Memory Bounded

Just as iterative deepening gives a more memory efficient version of BFS, can define IDA* as a more memory efficient version of A*.

Just use DFS with a cutoff on f values. Repeat with larger cutoff until solution found.

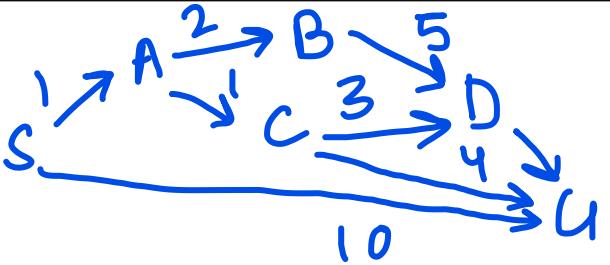
What to Learn?

The A* algorithm: its definition and behavior (finds optimal).

How to create admissible heuristics via relaxation.

A*

- Let $g(n)$ be cost incurred already on path to n
- Expand nodes with lowest $g(n) + h(n)$ first
- Say: $h(A) = 9, h(B) = 5, h(D) = 6, h(E) = 3, h(F) = 0$
- Note: if $h=0$ everywhere, then just uniform cost search



| S | A | B | C | D | G |
|---|---|---|---|---|---|
| 5 | 3 | 4 | 2 | 6 | 0 |

① Open = {A, G} Closed = {S}

$$f(n)_A = 3 + 1 = 4 \checkmark$$

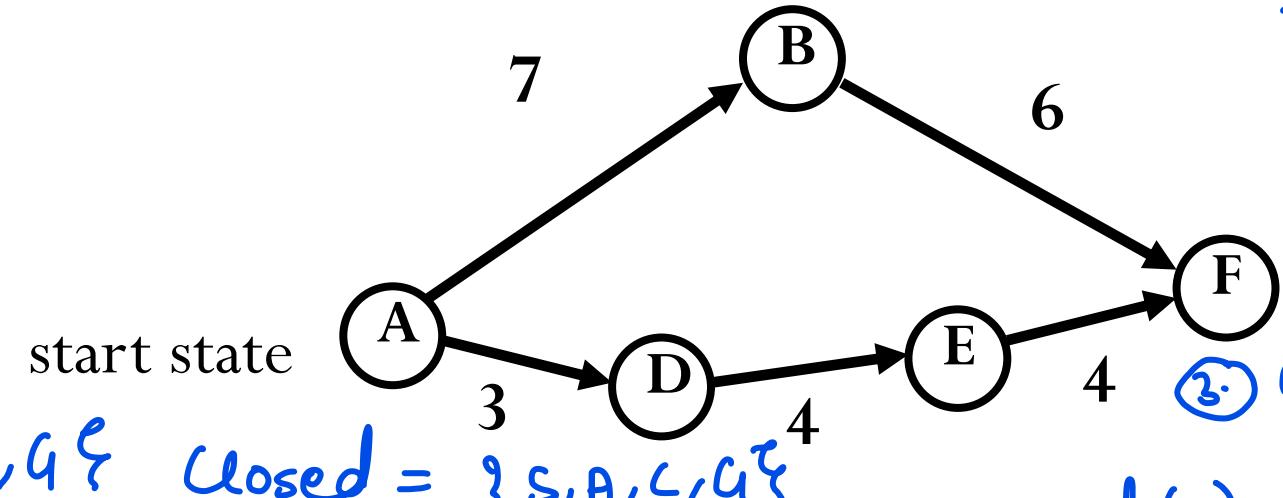
$$f(n)_G = 0 + 10 = 10$$

② Open = {B, C, G}

Closed = {S, A}

$$f(n)_B = 4 + 2 = 6$$

$$f(n)_C = 2 + 1 = 3 \checkmark$$



goal state

④ Open = {D, B, G} Closed = {S, A, C, G}

Ans $S \rightarrow A \rightarrow C \rightarrow G$

③ Open = {D, G, B, F}

Closed = {S, A, C}

$$f(n)_D = 6 + 3 = 9 \quad f(n)_G = 0 + 4 = 4 \checkmark$$

Admissibility

- A heuristic is **admissible**
 - if it never overestimates the distance to get the goal
 - If n is the optimal solution reachable from n' , then $g(n) \geq g(n') + h(n')$
- Straight-line distance is admissible: can't hope for anything better than a straight road to the goal
- Admissible heuristic means that A* is always optimistic
- A* is guaranteed to return a least-cost path from start to goal.

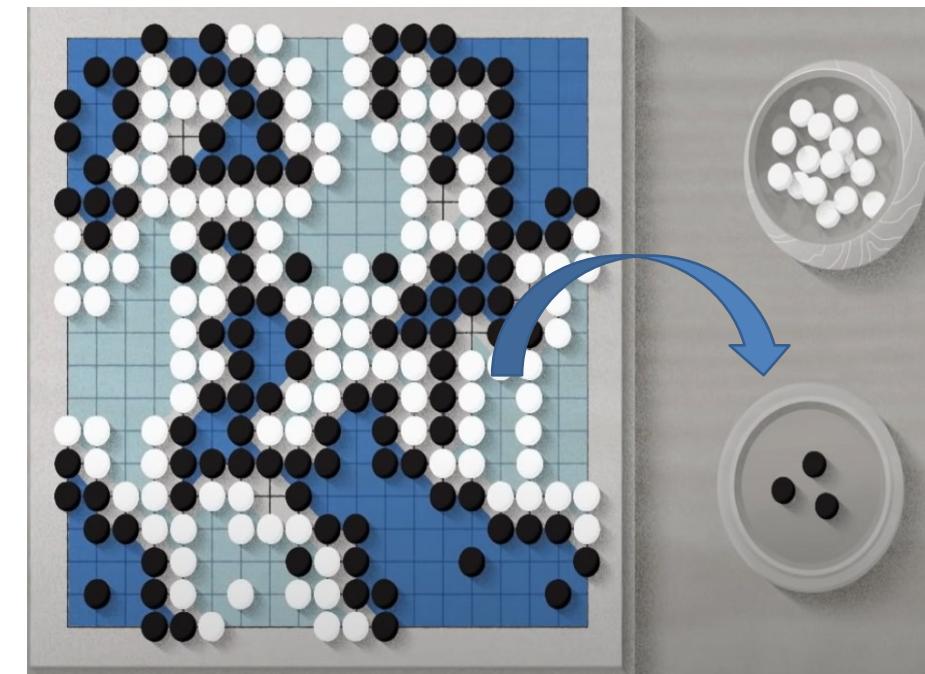
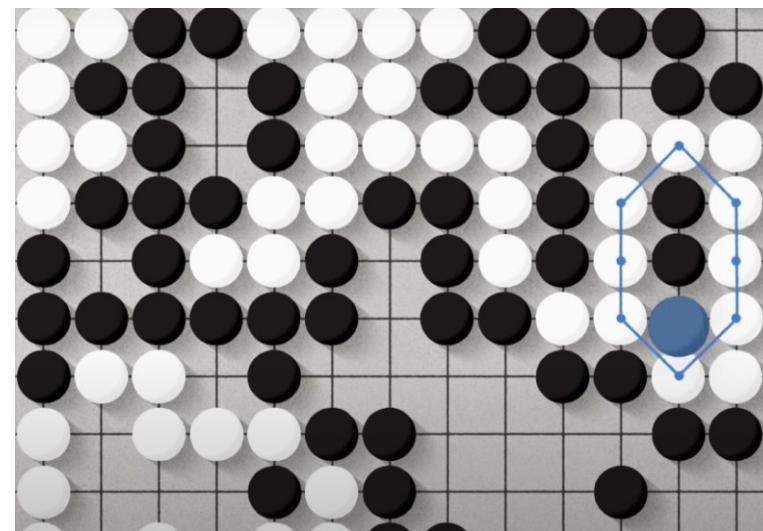
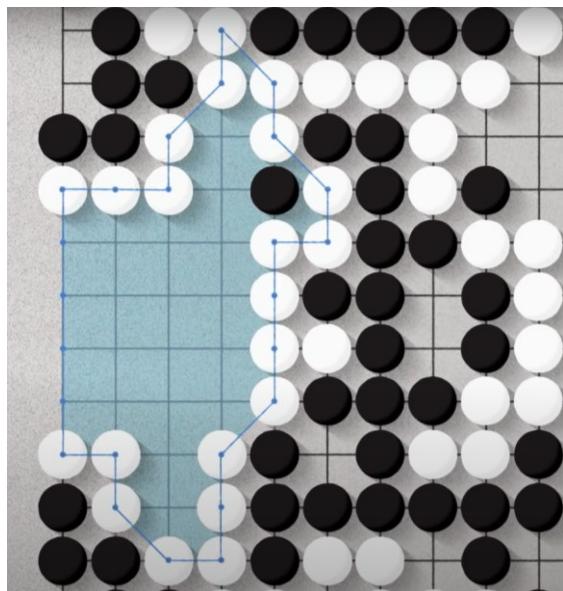
$h(n) \leq h^*(n)$ where $h(n)$: heuristic cost
 $h^*(n)$: expected cost

Alpha Go Searching Example

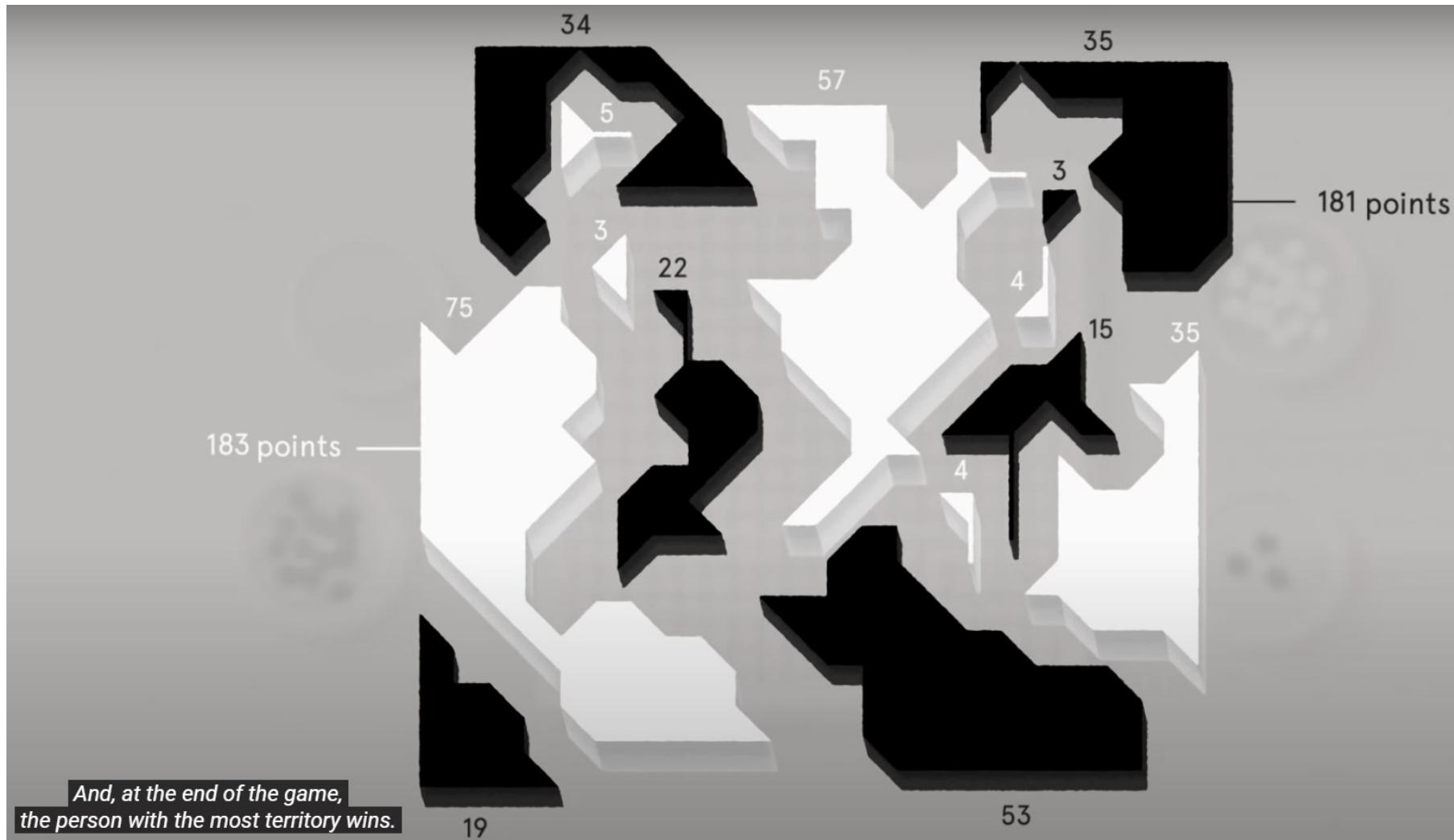
Go through RP note.

2016: Alpha Go

- Make territories
- Capture opponent pieces
- Largest territory

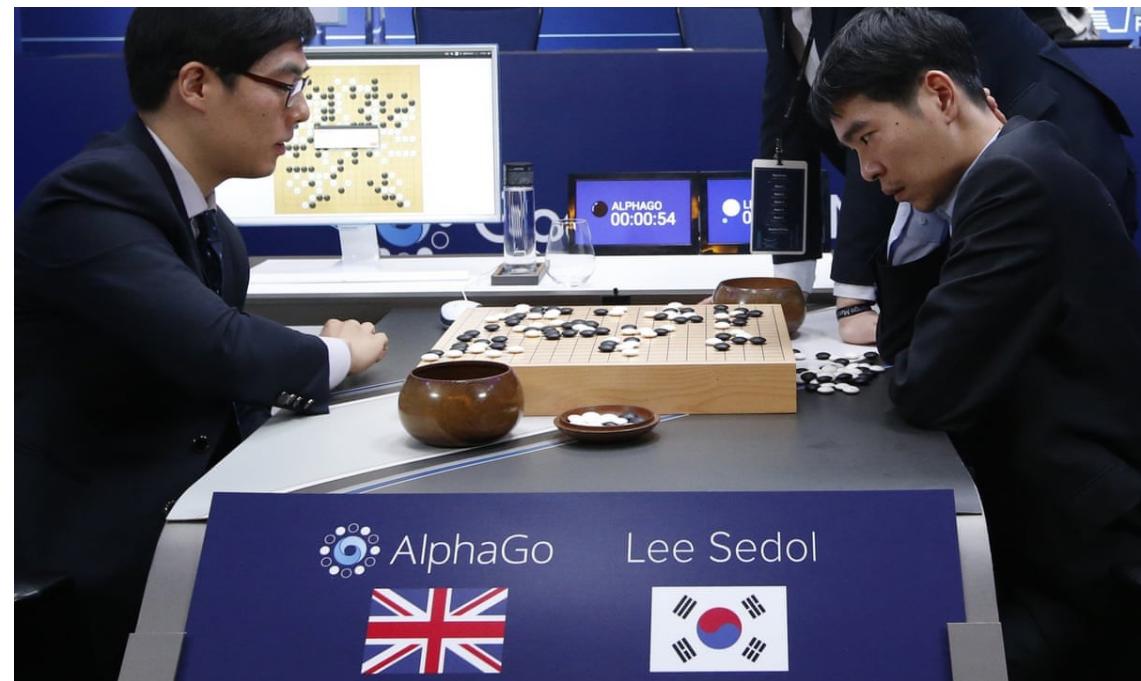


2016: Alpha Go

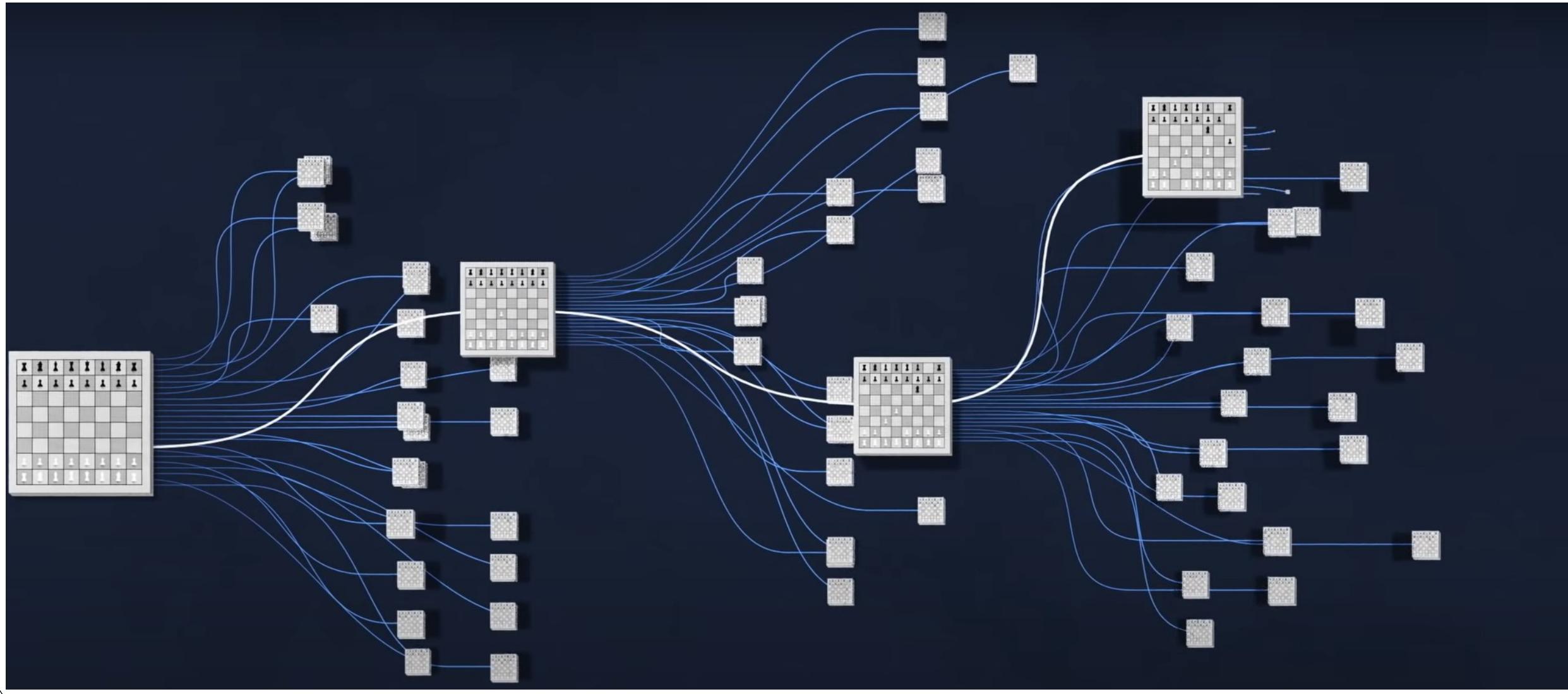


2016: Alpha Go

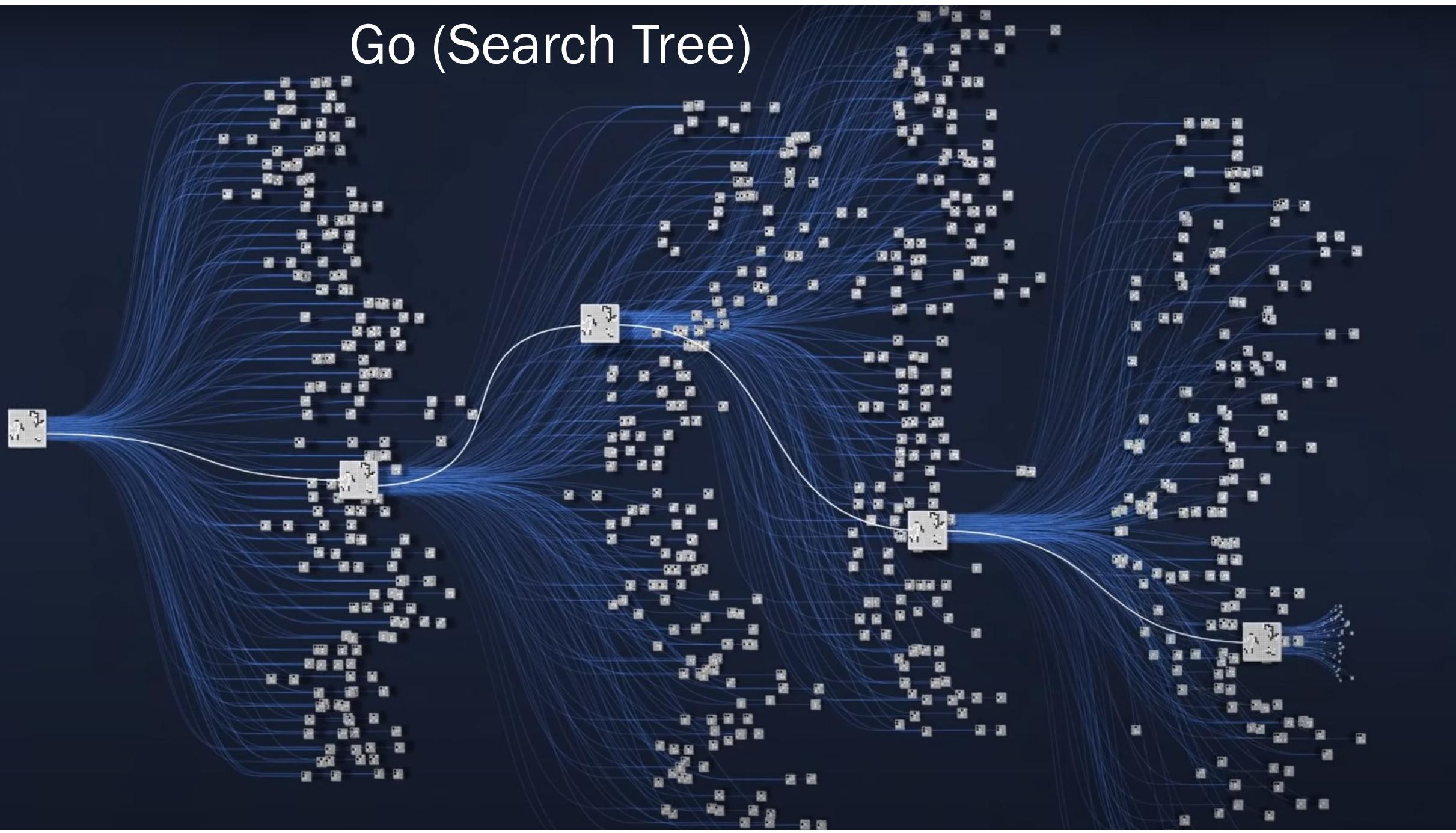
- <https://www.buzzfeednews.com/article/alexkantrowitz/were-in-an-artificial-intelligence-hype-cycle>



Chess (Search Tree)



Go (Search Tree)



Chess v/s Go

- At the opening move in Chess there are 20 possible moves.
- Go is simpler than Chess and yet more complex.
- In large games $O(b^d)$, such as
 - chess ($b \approx 35, d \approx 80$)
 - Go ($b \approx 250, d \approx 150$),
- Exhaustive search is infeasible, but the effective search space can be reduced by two general principles.
 - the depth of the search may be reduced by position evaluation: truncating (cut) the search tree at state s and replacing the subtree below s by an approximate value function $v(s) \approx v^*(s)$ that predicts the outcome from state s .
 - the breadth of the search may be reduced by sampling actions from a policy $p(a | s)$ that is a probability distribution over possible moves a in position s .

References

- Stuart Russel, and Peter Norvig. "Artificial intelligence: A modern approach. third edit." Upper Saddle River, New Jersey 7458 (2015).
- Introduction to Artificial Intelligence, Michael L. Littman, Fall 2001
mlittman@cs.brown.edu <https://courses.cs.duke.edu/fall08/cps270/>
- Vincent Conitzer, Artificial Intelligence <http://www.cs.duke.edu/courses/fall08/>
- Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." Nature 529.7587 (2016): 484-489.
- AlphaGo - The Movie | Full award-winning documentary "DeepMind"
<https://www.youtube.com/watch?v=WXuK6gekU1Y>

ឃុំបរិញ្ញា

धन्यवादः
Sanskrit

Ευχαριστώ^ω
Greek

Спасибо
Russian

شُكْرًا

Arabic

多謝

Traditional
Chinese

多谢

Simplified
Chinese

ありがとうございました

Japanese

Grazie
Italian

תודה רבה
Hebrew

ಧನ್ಯವಾದಗಳು
Kannada

Thank You
English

Gracias
Spanish

Obrigado
Portuguese

धन्यवाद

Hindi

Danke
German

நன்றி

Tamil
Tamil

감사합니다

Korean

<https://sites.google.com/site/animeshchaturvedi07>