Artificial Intelligent – Def. , History & Search

Dr Salem Ameen

S.A.AMEEN1@SALFORD.AC.UK

Contents

Maximum

Minimum

Preface (pdf); Contents with subsections

I Artificial Intelligence
1 Introduction ... 1
2 Intelligent Agents ... 36

II Problem-solving
3 Solving Problems by Searching ... 63
4 Search in Complex Environments ... 110
5 Adversarial Search and Games ... 146
6 Constraint Satisfaction Problems ... 180

III Knowledge, reasoning, and planning
7 Logical Agents ... 208
8 First-Order Logic ... 251
9 Inference in First-Order Logic ... 280
10 Knowledge Representation ... 314
11 Automated Planning ... 344

IV Uncertain knowledge and reasoning
12 Quantifying Uncertainty ... 385
13 Probabilistic Reasoning ... 412
14 Probabilistic Reasoning over Time ... 461
15 Probabilistic Programming ... 500
16 Making Simple Decisions ... 528
17 Making Complex Decisions ... 562
18 Multiagent Decision Making ... 599

V Machine Learning
19 Learning from Examples ... 651
20 Learning Probabilistic Models ... 721
21 Deep Learning ... 750
22 Reinforcement Learning ... 789

VI Communicating, perceiving, and acting
23 Natural Language Processing ... 823
24 Deep Learning for Natural Language Processing ... 856
25 Computer Vision ... 881
26 Robotics ... 925

VII Conclusions
27 Philosophy, Ethics, and Safety of AI ... 981
28 The Future of AI ... 1012
Appendix A: Mathematical Background ... 1023
Appendix B: Notes on Languages and Algorithms ... 1030
Bibliography ... 1033 (pdf and bib data)
Index ... 1069 (pdf)

Exercises (website)
Figures (pdf)
Code (website); Pseudocode (pdf)

Lecture 1: Definition of AI

- "Intelligence: The ability to learn and solve problems" Webster's Dictionary.
- "Artificial intelligence (AI) is the intelligence exhibited by machines or software' Wikipedia.
- "The science and engineering of making intelligent machines" McCarthy.
- "The study and design of intelligent agents, where an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success." Russel & Norvig AI book

Lecture 1: Applications of AI

Speech recognition • Autonomous planning and scheduling • Financial forecasting • Game playing, video games • Spam fighting • Logistics planning • Robotics (household, surgery, navigation) • Machine translation • Information extraction • VLSI layout • Automatic asse • Sentiment analysis • Fraud detection • Recommendation systems • Web search engines • Autonomous cars • Energy optimization • Question answering systems • Social network analysis • Medical diagnosis, imaging • Route finding • Traveling salesperson • Protein design • Document summarization • Transportation/scheduling • Computer animation

Lecture 1: AI founders

- Aristotle
- Alan Turing
- John Mc Carthy
- Warren McCulloh
- Walter Pitts
- Claude Shannon
- Marvin Minsky
- Dean Edmonds
- Herbert Simon
- Allen Newell
- David Waltz
- Tom Mitchell
- Stuart J. Russell
- Peter Norvig
- etc …

Lecture 1: AI Resources

Major journals/conferences: JAIR, TPAMI, JMLR, IJCAI, AAAI, IAAI,

CVPR, ECAI, ICML, NIPS, etc.

Select History of Artificial Intelligence

1950 | 1960 | 1970 | 1980 | 1990 | 2000 | 2010 | 2017

AI Winters 1974–1980 and 1987–1993

1950 - Computing Machinery and Intelligence "Turing Test" published by MIND vol. LIX

1956 - Dartmouth Summer Research Project on AI

J. McCarthy, M. Minsky, N. Rochester, O. Selfridge, C. Shannon, others

1955 - Western Joint Computer Conference Session on Learning Machines

W.A. Clark
Generalization of Pattern Recognition in a Self-Organizing System (MIT LL Staff)

G. Dineen
Programming Pattern Recognition (MIT LL Staff)

O. Selfridge
Pattern Recognition and Modern Computers (MIT LL Staff)

1957 - Frank Rosenblatt Neural Networks Perceiving and Recognizing Automation

1957 - Memory Test Computer, first computer to simulate the operation of neural networks

1958 - National Physical Laboratory in the UK Symposium on the Mechanization of Thought Processes

1959 - Arthur Samuel "Some studies in machine learning using the Game of checkers" IBM Journal of R&D

1960 - Recognizing hand-written characters, Robert Larson of SRI AI Center

1961 - James Slagle, Solving Freshman Calculus (Minsky Student) MIT

J. Forgie and J. Allen

1979 - An Assessment of AI from a Lincoln Laboratory Perspective Internal MIT LL publication

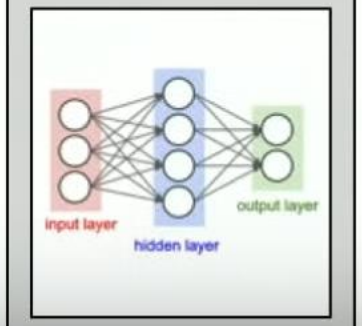
1982 - Expert Systems Pioneer DENDRAL project at Stanford

1984 - Hidden Markov models

1988 - Statistical Machine Translation

1989 - Convolutional Neural Networks

1986–present
The return of neural networks



1994 - Human-level spontaneous speech recognition

1997 - IBM Deep Blue defeats reigning chess champion *(Garry Kasparov)*

2001–present
The availability of very large data sets

2005 - Google's Arabic and Chinese to English translation

2007 - DARPA Grand Challenge *("Urban Challenge")*

2011 - IBM Watson defeats former *Jeopardy!* champions *(Brad Rutter and Ken Jennings)*

2012 - Team from U. of Toronto (Geoff Hinton's lab) wins the ImageNet Large Scale Visual Recognition Challenge with deep-learning software

2014 - Google's GoogleNet Object classification at near human performance

2015 - DeepMind achieves human expert level of play on Atari games *(using only raw pixels and scores)*

2016 - DeepMind AlphaGo defeats top human Go player *(Lee Sedol)*

2016 - DARPA Cyber Grand Challenge

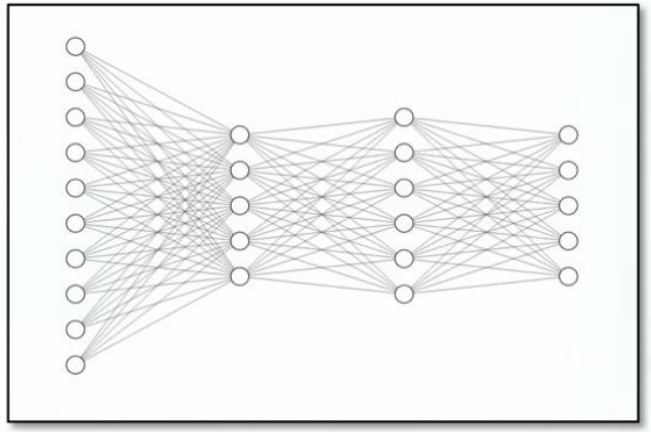
Lecture 1: History of AI

AI. Why Now?

Big Data

Compute Power

Machine Learning Algorithms



Course logistics

- **Course level**: Master
- **Prerequisites:** You are required to have some knowledge of programming and an understanding of probability. Python is the programming language in this course.
- **Assessments**: There will be two kinds of assessments :
  - Research + presentation :
  - Projects (programming):

Lecture 1: AI Paradigm

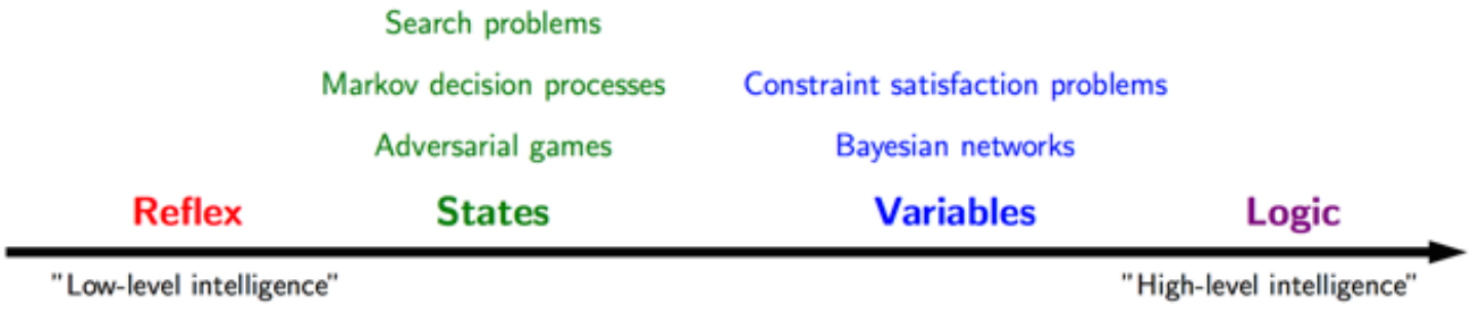
Modeling

Inference **Learning**

Lecture 1: Agent

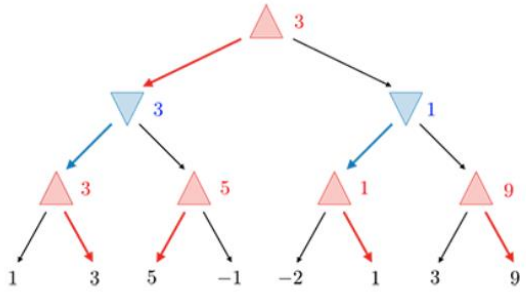
- An entity that **perceives** its **environment** and **acts** upon that environment.
- In a navigator app, for example, the agent would be a representation of a car that needs to decide on which actions to take to arrive at the destination.

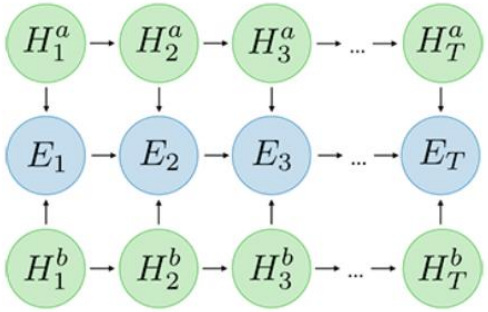
Lecture 1: Agent Types

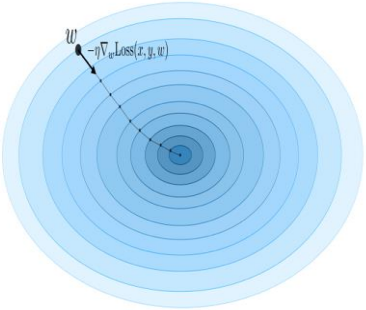


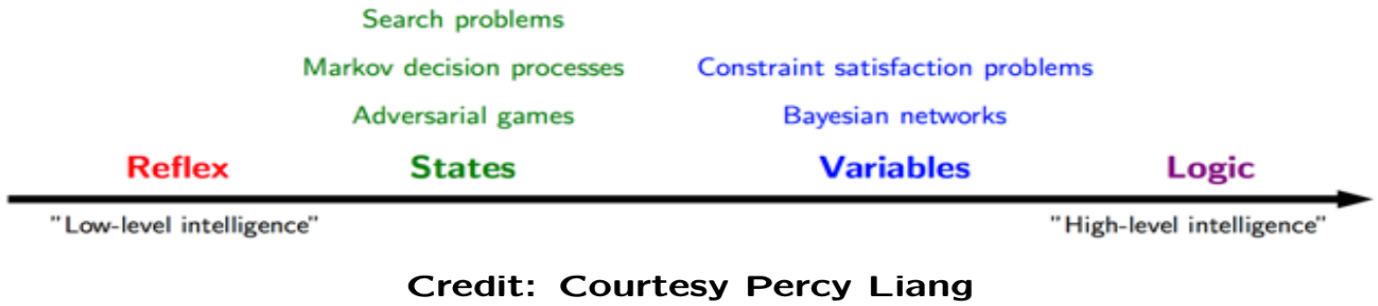
Credit: Courtesy Percy Liang

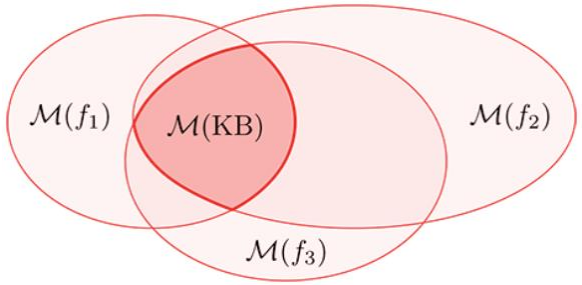
Lecture 1: Agent Types











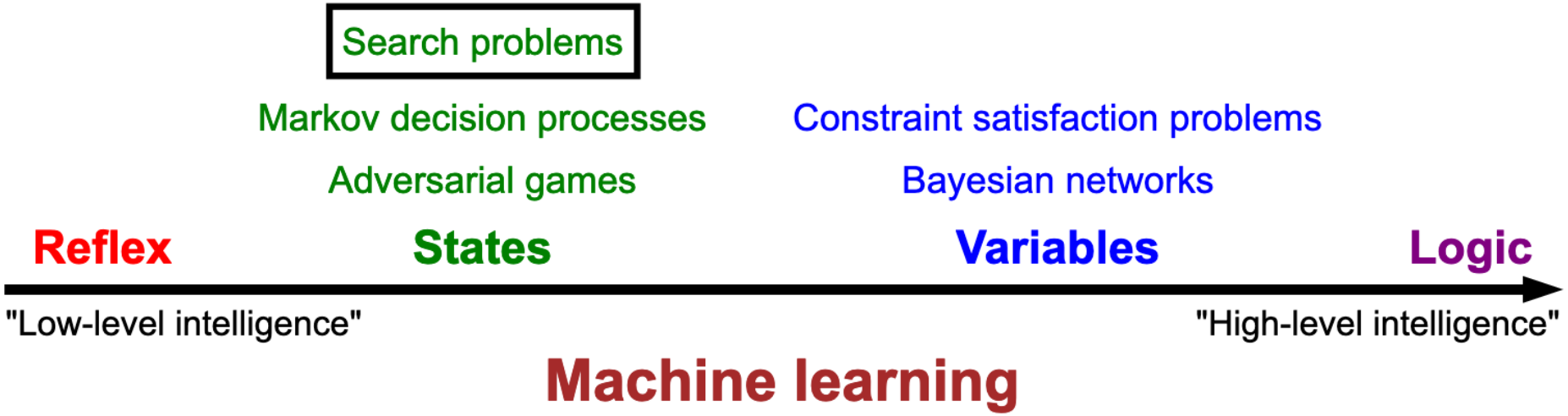
Credit: Courtesy Percy Liang

Lecture 1: Search

- Finding a solution to a problem, like a **navigator app** that finds the best route from your origin to the destination, or like **playing a game** and figuring out the next move.
- A navigator app uses a typical search process, where the agent (the thinking part of the program) receives as input your current location and your desired destination, and, based on a search algorithm, returns a suggested path.
- However, there are many other forms of search problems, like puzzles or mazes.

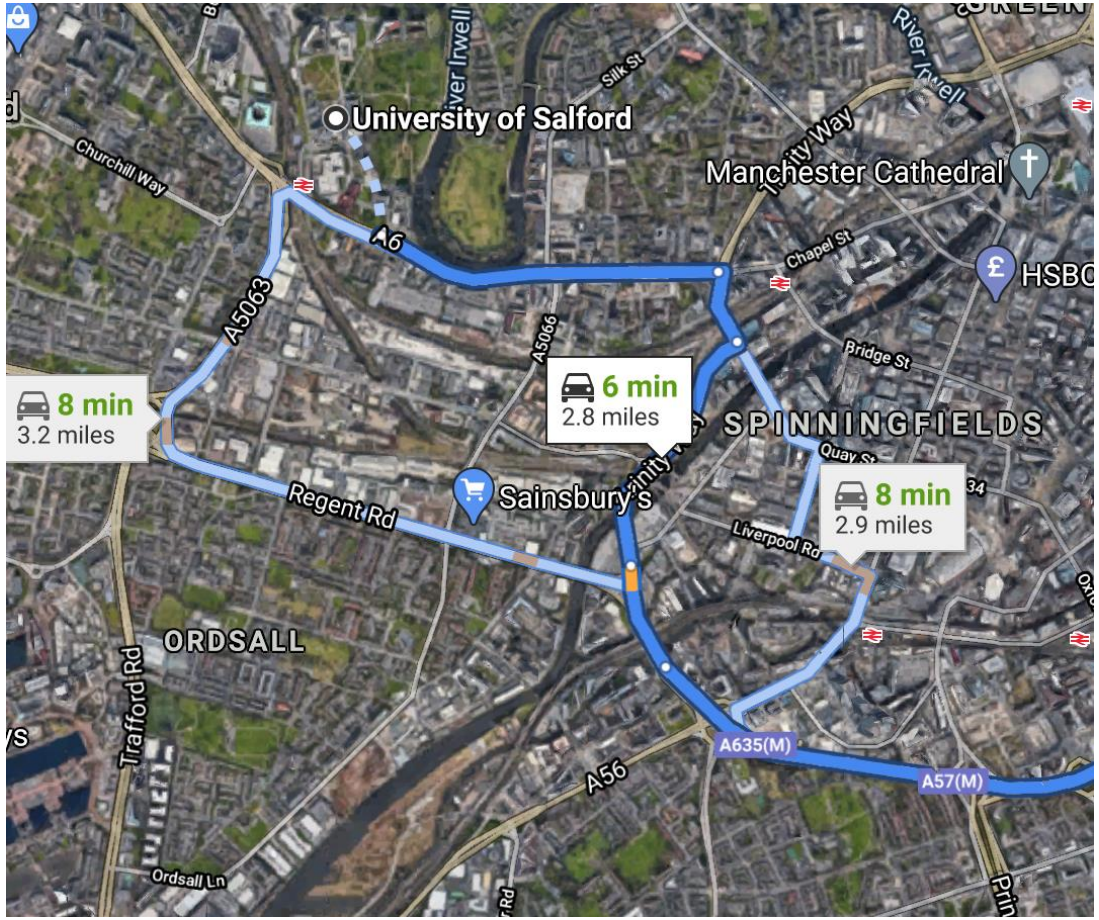
Lecture 1: Search

Search is not about map search is about choices



Lecture 1: Search
Application: route finding

- **Objective**: shortest? fastest? most scenic?
- **Actions**: go straight, turn left, turn right



Lecture 1: Search
Application: Gaming/Puzzles

- **Objective**: reach a certain configuration
- **Actions**: move

Lecture 1: Search
Application: machine translation

- **Objective**: fluent English and preserves meaning
- **Actions**: append single words (e.g., the)

la maison bleue

↓

the blue house

Lecture 1: Search

Search problem (state-based models):

$$x \longrightarrow \boxed{f} \longrightarrow \textbf{action sequence } (a_1, a_2, a_3, a_4, \ldots)$$

Lecture 1: Search
Initial State

Initial Start or Starting state

- The state from which the search algorithm starts.
- In a navigator app, that would be the current location.

Lecture 1: Search
Actions

- Choices that can be made in a state.

- Possible actions available to the agent. At a state s, Actions(s) returns the set of actions that can be executed in state s. (Action space)

Lecture 1: Search
Transition Model

Transition Model or Successor

- A description of what state results from performing any applicable action in any state.
- More precisely, the transition model can be defined as a function. Upon receiving state s and action a as input, ***Results(s, a***) returns the state resulting from performing action a in state s. ***Succ(s,a)***

Lecture 1: Search
State Space / States

- The set of all states reachable from the initial state by any sequence of actions.

Lecture 1: Search
Goal Test

- The condition that determines whether a given state is a goal state.
- IsEnd(s)
- In a navigator app, the goal test would be whether the current location of the agent (the representation of the car) is at the destination.
  - If it is — problem solved.
  - If it’s not — we continue searching.

Lecture 1: Search
Path Test

- Function that assigns a numeric cost to a path w.r.t. performance measure
- For example, a navigator app does not simply bring you to your goal; it does so while minimizing the path cost, finding the fastest way possible for you to get to your goal state.

Lecture 1: Search examples

- **States**: Location of each of the 8 tiles in the 3x3 grid .
- **Initial state:** Any state
- **Actions:** Move Left, Right, Up or Down
- **Transition model:** Given a state and an action, returns resulting state
- **Goal test**: state matches the goal state?
- **Path cost**: total moves, each move costs 1.

Lecture 1: Search
Solving Search Problems/ Solution

- **Solution**

A sequence of actions that leads from the initial state to the goal state.

- **Optimal Solution**
  - A solution that has the lowest path cost among all solutions.

Lecture 1: Search
Search strategies

- A strategy is defined by picking the order of node expansion
- Strategies are evaluated along the following dimensions:
  - Completeness
    Does it always find a solution if one exists?
  - Time complexity
    Number of nodes generated/expanded
  - Space complexity
    Maximum number of nodes in memory
  - Optimality
    Does it always find a least-cost solution?

Lecture 1: Search
Search strategies

- Time and space complexity are measured in terms of:
  - b: maximum branching factor of the search tree (actions per state).
  - d: depth of the solution
  - m: maximum depth of the state space (may be ∞) (also noted sometimes D).

- Two kinds of search: Uninformed and Informed.

Lecture 1: Search
Solving Search Problems/Node

In a search process, data is often stored in a **node**, a ***data structure*** *(hold information)* that contains the following data:

- A **state**
- Its ***parent node***, through which the current node was generated
- The ***action*** that was applied to the state of the parent to get to the current node
- The ***path cost*** from the initial state to this node

Lecture 1: Search
Solving Search Problems/ frontier

Search algorithms

Uninformed - Use no domain knowledge!

1. Depth-first search (DFS): Expand deepest node
2. Breadth-first search (BFS): Expand shallowest node
3. Depth-limited search (DLS): Depth first with depth limit
4. Iterative-deepening search (IDS): DLS with increasing limit
5. Uniform-cost search (UCS): Expand least cost node

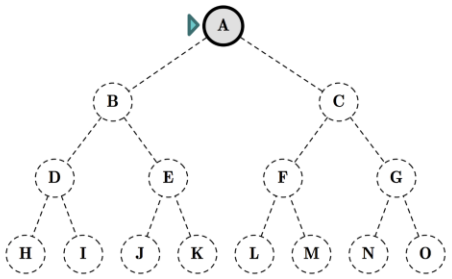
Informed

Lecture 1: Search
Depth-First Search (DFS)

Assume action costs $\mathrm{Cost}(s, a) = 0$.

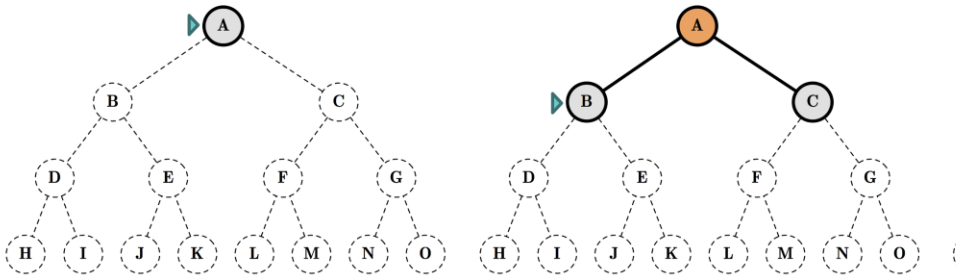
Lecture 1: Search
Depth-First Search (DFS)

DFS:
Expand deepest first.



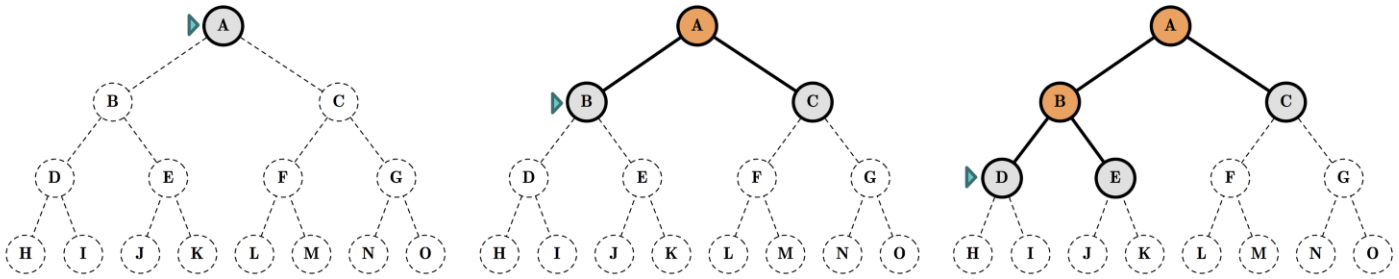
Lecture 1: Search
Depth-First Search (DFS)

DFS:
Expand deepest first.



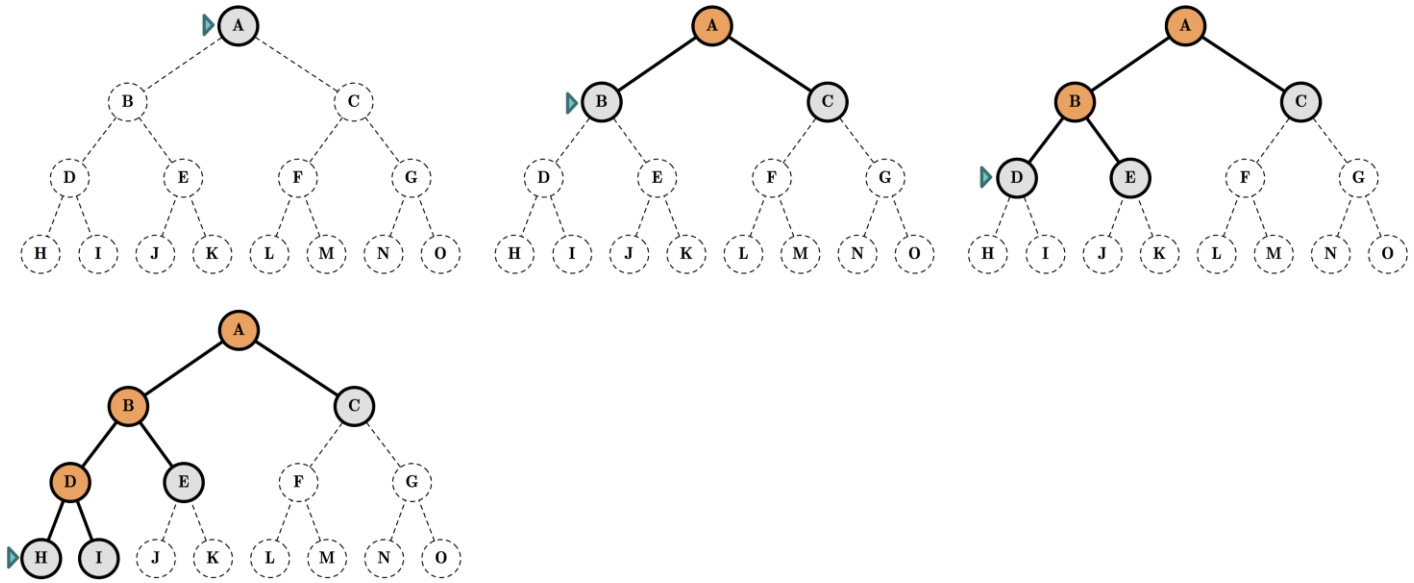
Lecture 1: Search
Depth-First Search (DFS)

DFS:
Expand deepest first.



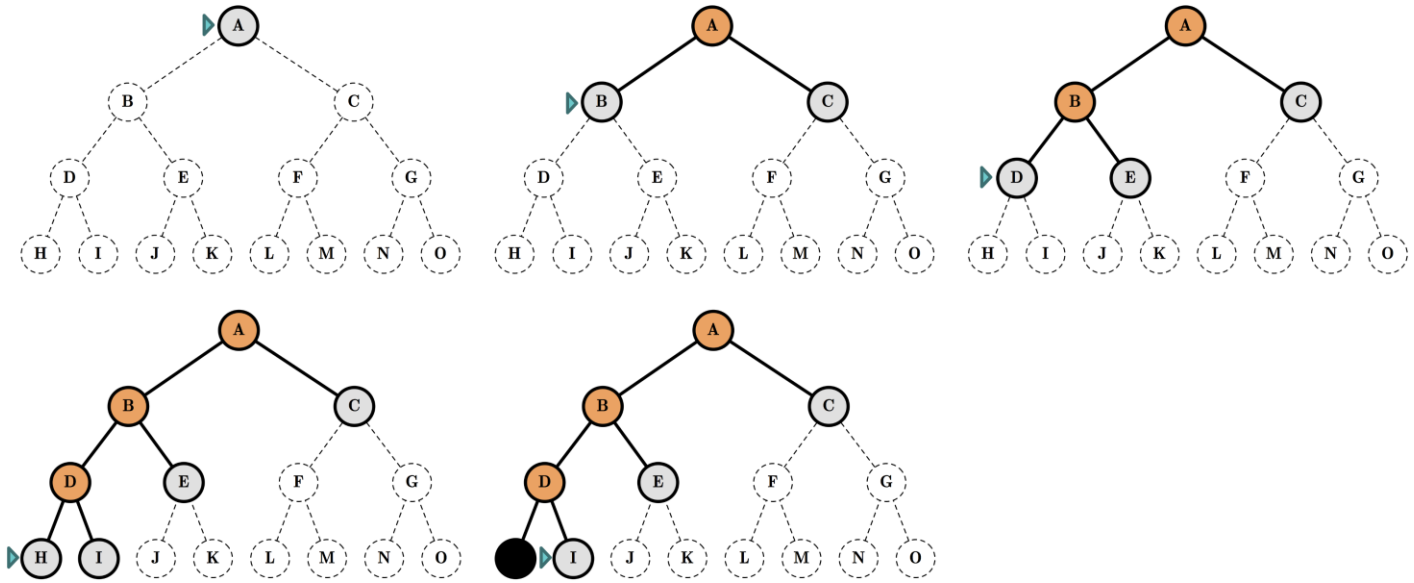
Lecture 1: Search
Depth-First Search (DFS)

DFS:
Expand deepest first.



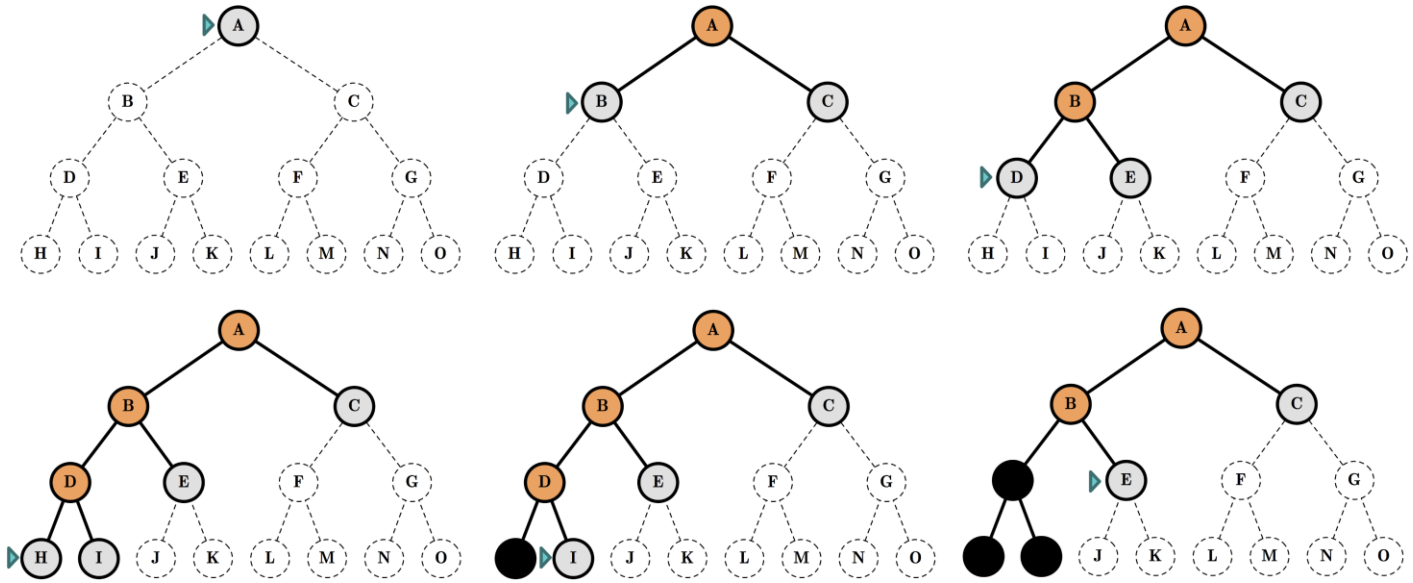
Lecture 1: Search
Depth-First Search (DFS)

DFS:
Expand deepest first.



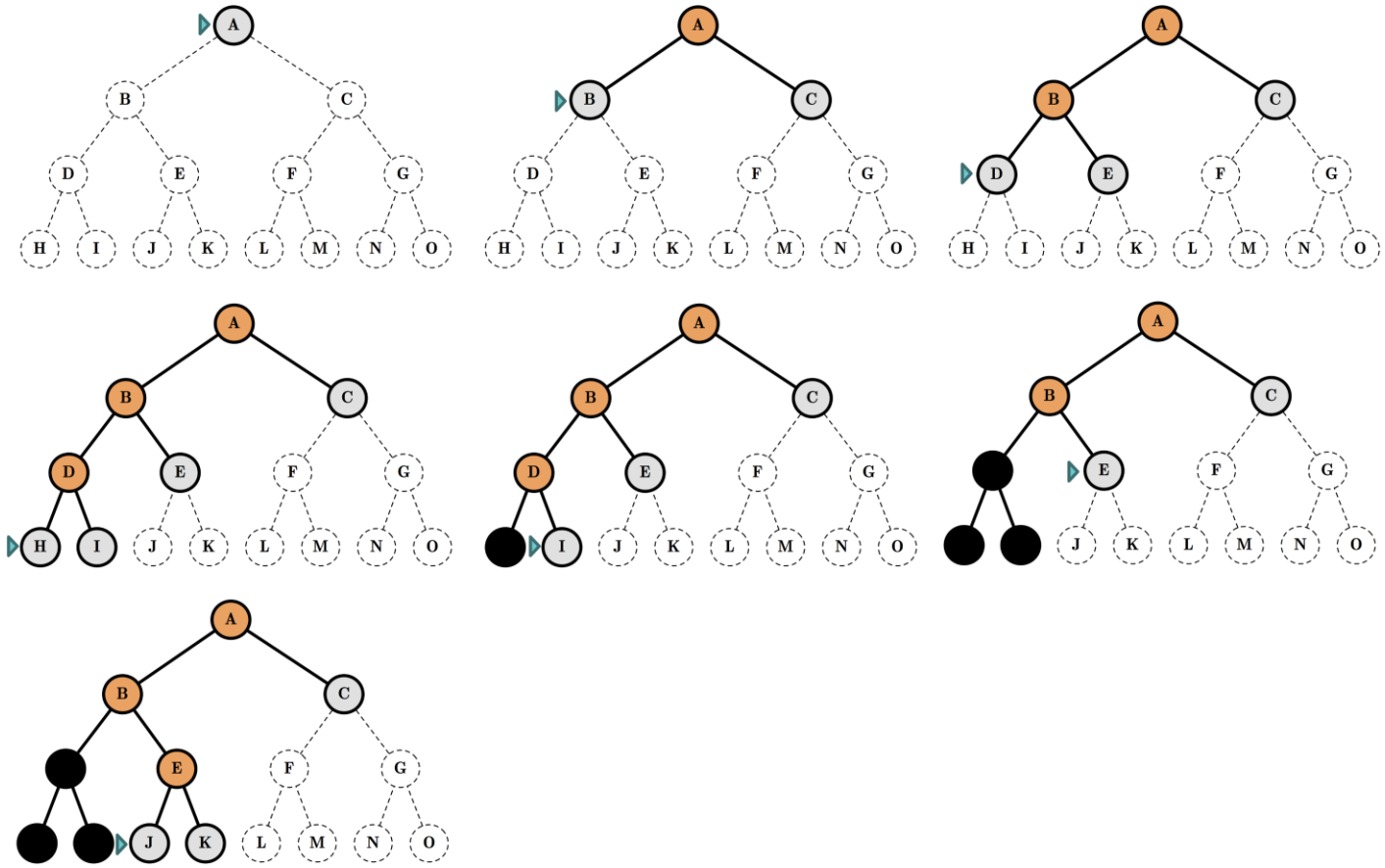
Lecture 1: Search
Depth-First Search (DFS)

DFS:
Expand deepest first.



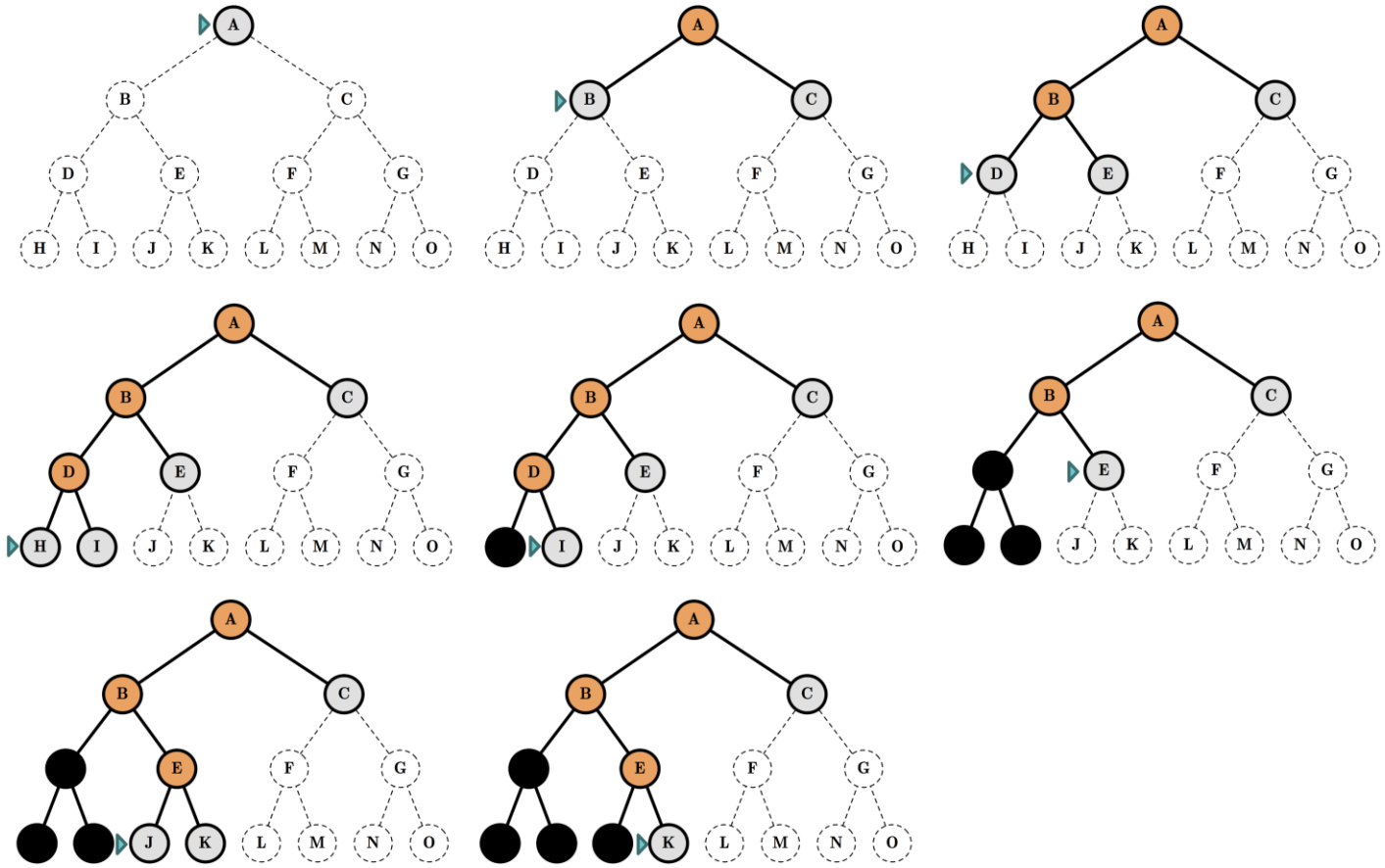
Lecture 1: Search
Depth-First Search (DFS)

DFS:
Expand deepest first.



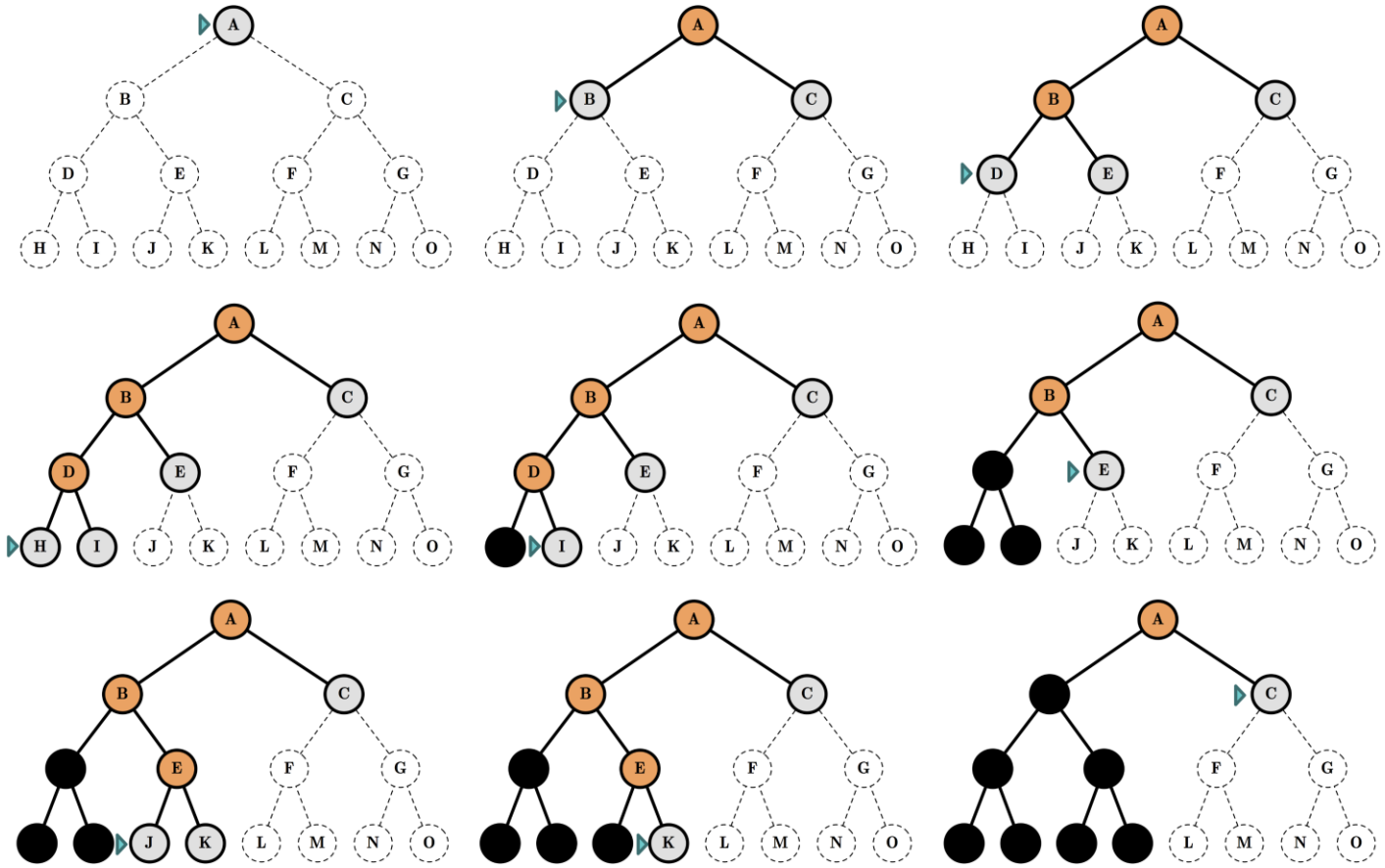
Lecture 1: Search
Depth-First Search (DFS)

DFS:
Expand deepest first.



Lecture 1: Search
Depth-First Search (DFS)

DFS:
Expand deepest first.



Lecture 1: Search
Depth-First Search (DFS)

- **Complete** No: fails in infinite-depth spaces, spaces with loops
  Modify to avoid repeated states along path.
  $\Rightarrow$ complete in finite spaces
- **Time** $O(b^m)$: $1 + b + b^2 + b^3 + \ldots + b^m = O(b^m)$
  bad if m is much larger than d
  but if solutions are dense, may be much faster than BFS.
- **Space** $O(bm)$ linear space complexity! (needs to store only a single path from the root to a leaf node, **along with the remaining unexpanded sibling nodes for each node on the path, hence the m factor**.)
- **Optimal** No
- **Implementation:** fringe: LIFO (Stack)

Lecture 1: Search
Depth-First Search (DFS)

University of Salford MANCHESTER

Pros:

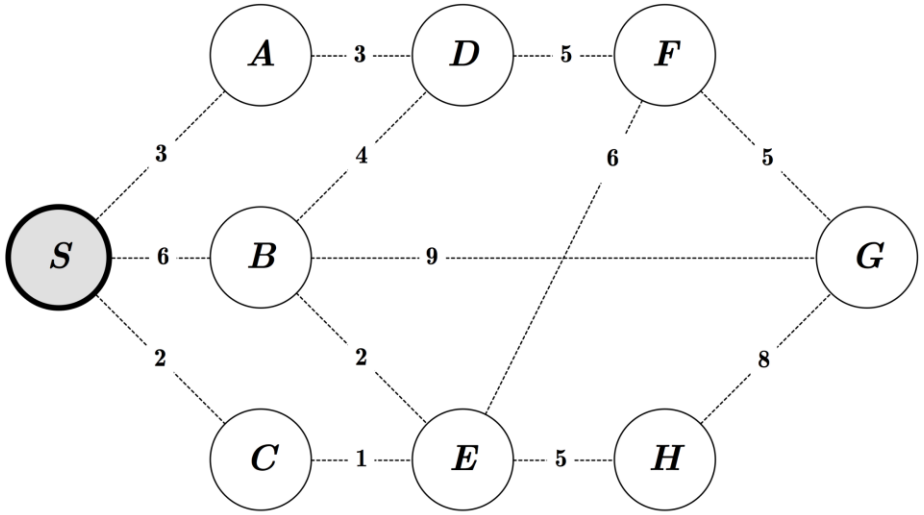
- At best, this algorithm is the fastest. If it “lucks out” and always chooses the right path to the solution (by chance), then depth-first search takes the least possible time to get to a solution.

Cons:

- It is possible that the found solution is not optimal.
- At worst, this algorithm will explore every possible path before finding the solution, thus taking the longest possible time before reaching the solution.

Lecture 1: Search
DFS Example

DFS:
Expand deepest first.
Checked

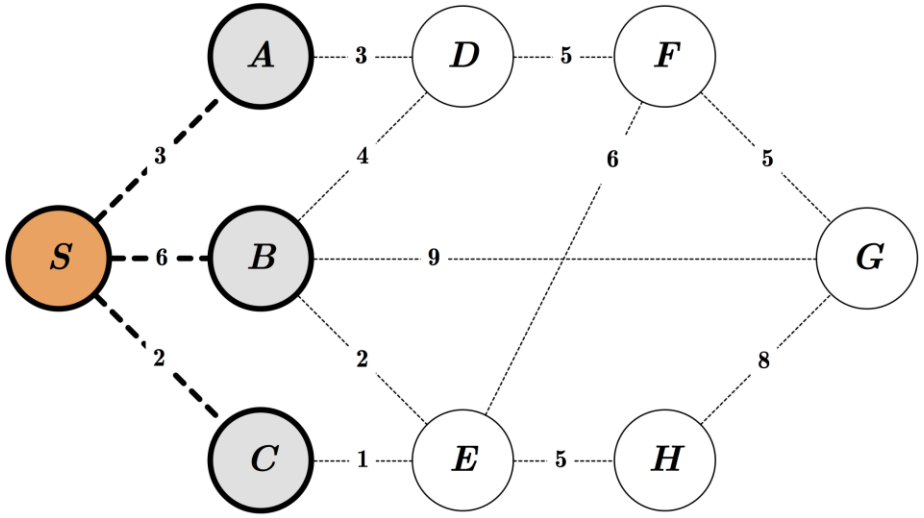


Stack:

S

Order of Visit:

Lecture 1: Search
DFS Example



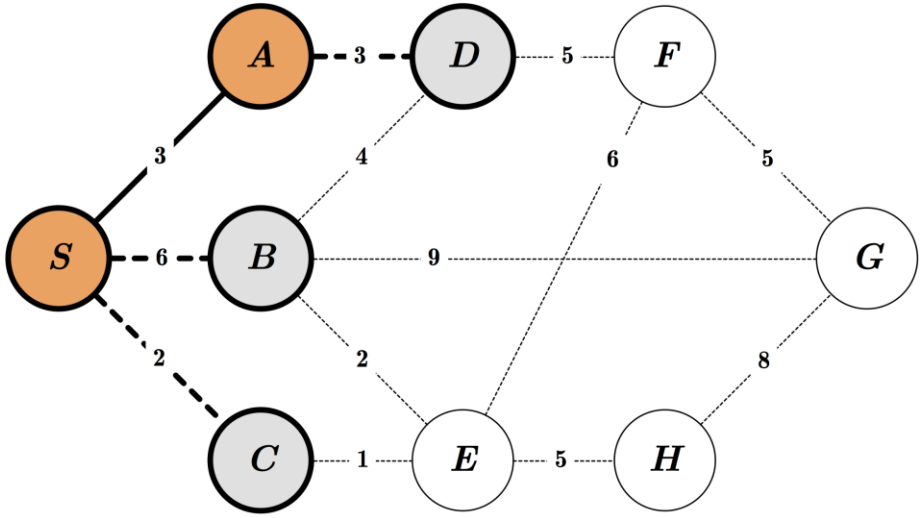
Stack:

S	*C*	*B*	*A*

Order of Visit:

S

Lecture 1: Search
DFS Example



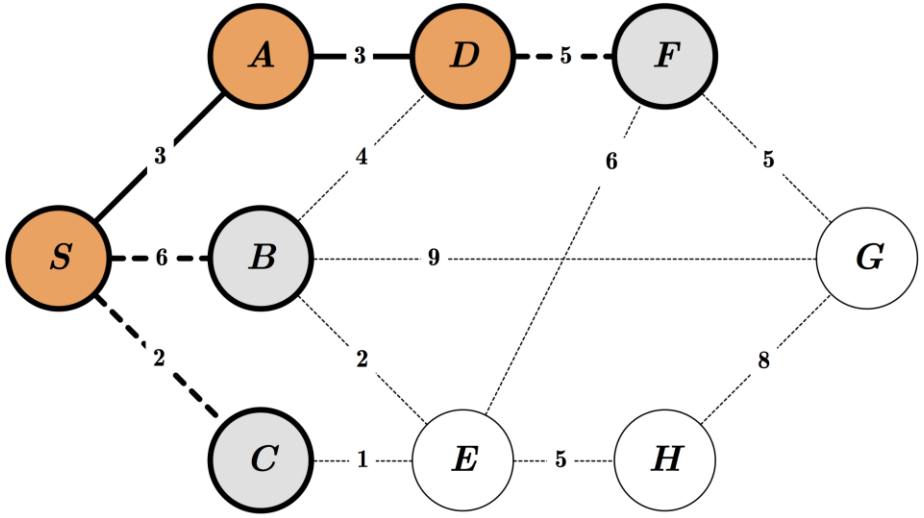
Stack:

S	*C*	*B*	*A*	*D*

Order of Visit:

S *A*

Lecture 1: Search
DFS Example



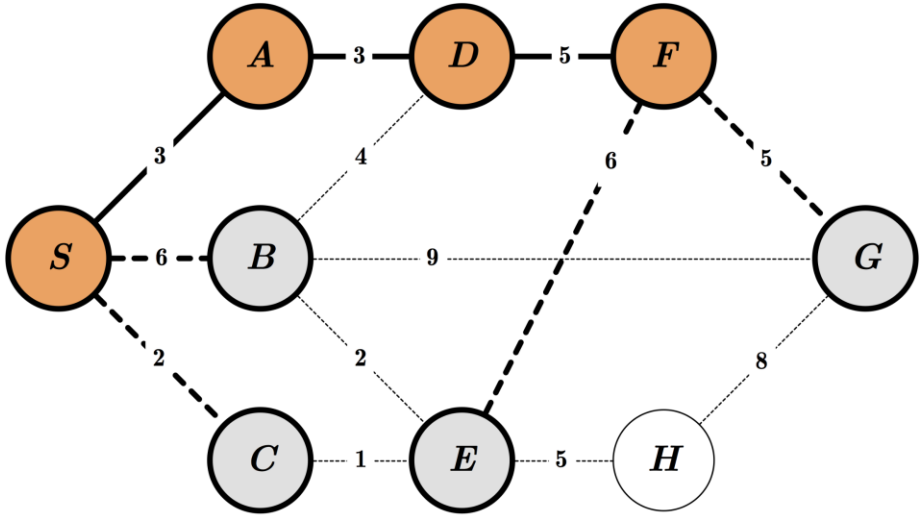
Stack:

S	*C*	*B*	*A*	*D*	*F*

Order of Visit:

S *A* *D*

Lecture 1: Search
DFS Example



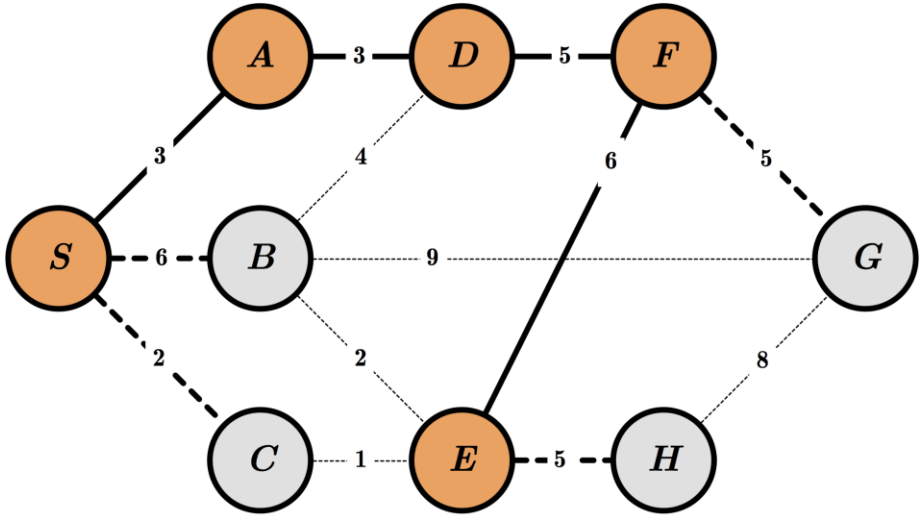
Stack:

S	*C*	*B*	*A*	*D*	*F*	*G*	*E*

Order of Visit:

S *A* *D* *F*

Lecture 1: Search
DFS Example



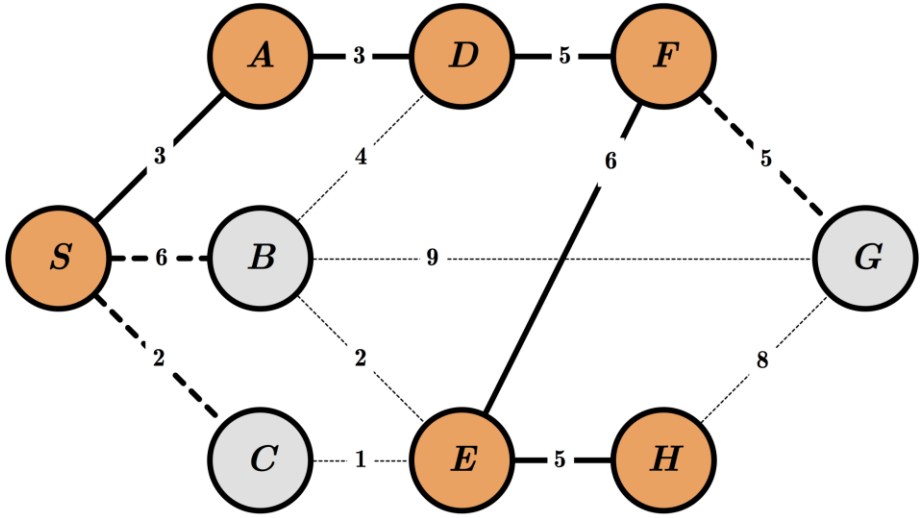
Stack:

S	*C*	*B*	*A*	*D*	*F*	*G*	*E*	*H*

Order of Visit:

S *A* *D* *F* *E*

Lecture 1: Search
DFS Example



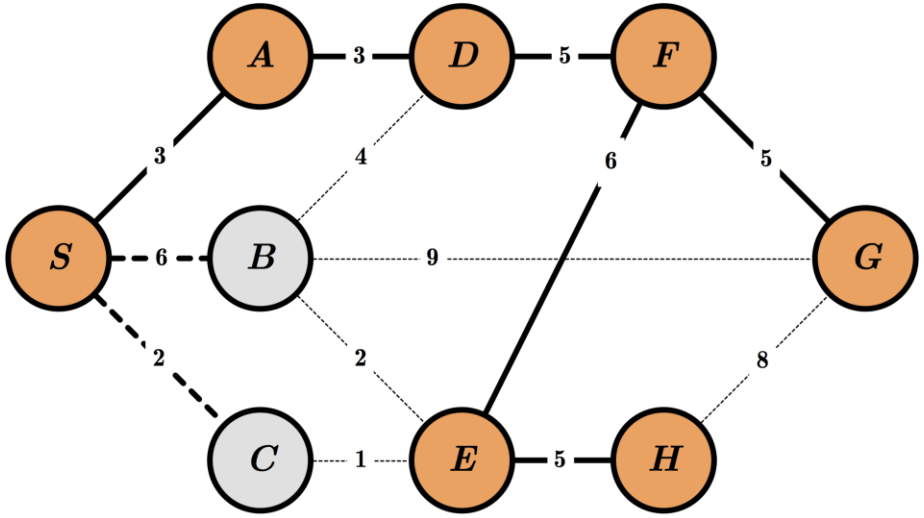
Stack:

S	*C*	*B*	*A*	*D*	*F*	*G*	*E*	*H*

Order of Visit:

S *A* *D* *F* *E* *H*

Lecture 1: Search
DFS Example



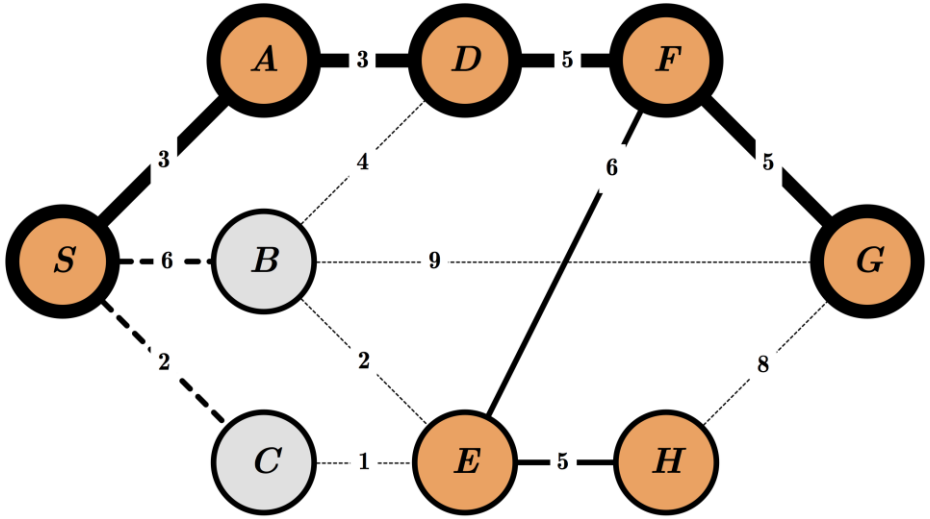
Stack:

S	*C*	*B*	*A*	*D*	*F*	*G*	*E*	*H*

Order of Visit:

S *A* *D* *F* *E* *H* *G*

Lecture 1: Search
DFS Example



Stack:

S	*C*	*B*	*A*	*D*	*F*	*G*	*E*	*H*

Order of Visit:

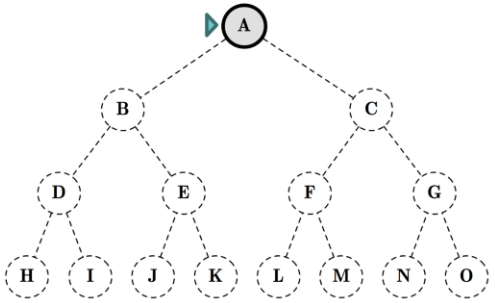
S *A* *D* *F* *E* *H* *G*

Lecture 1: Search
Breadth-First Search (BFS)

Assume action costs $\mathrm{Cost}(s, a) = c$ for some $c \geq 0$.

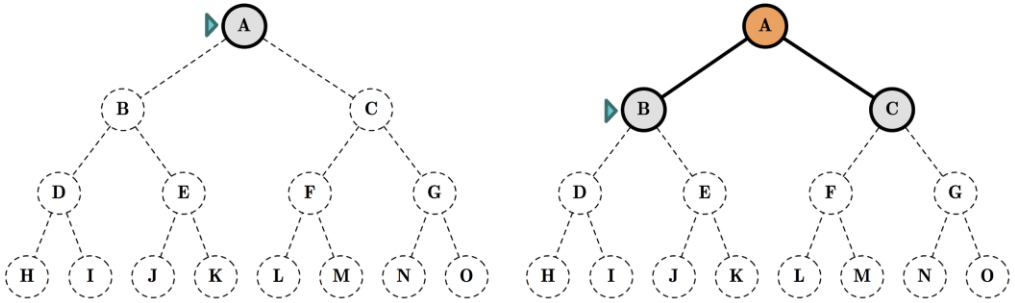
Lecture 1: Search
Breadth-First Search (BFS)

BFS:
Expand shallowest first.



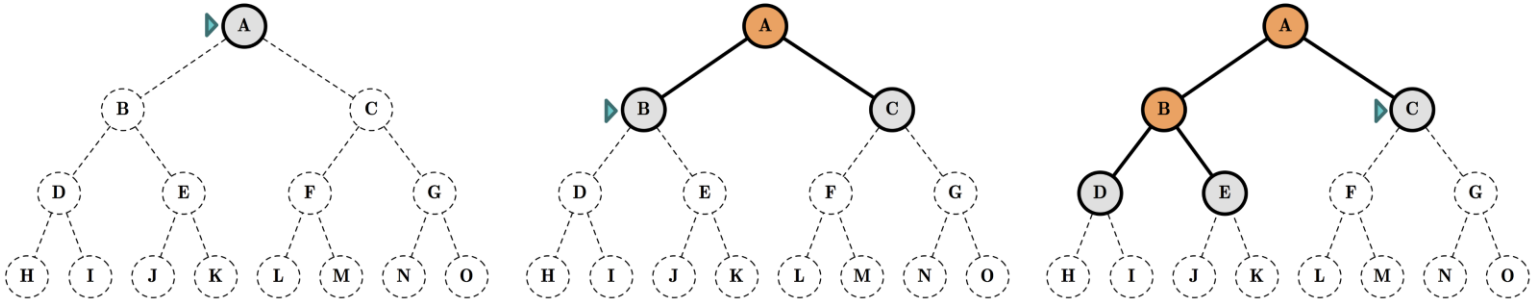
Lecture 1: Search
Breadth-First Search (BFS)

BFS:
Expand shallowest first.



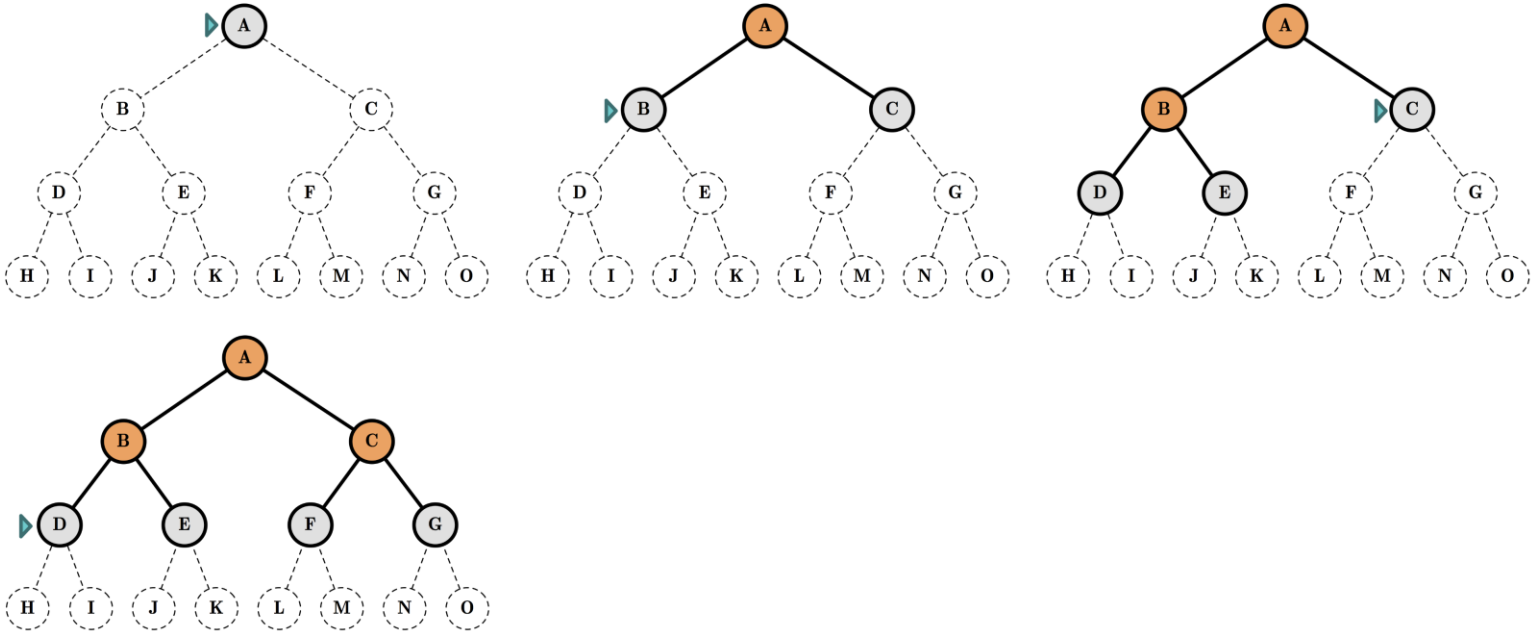
Lecture 1: Search
Breadth-First Search (BFS)

BFS:
Expand shallowest first.



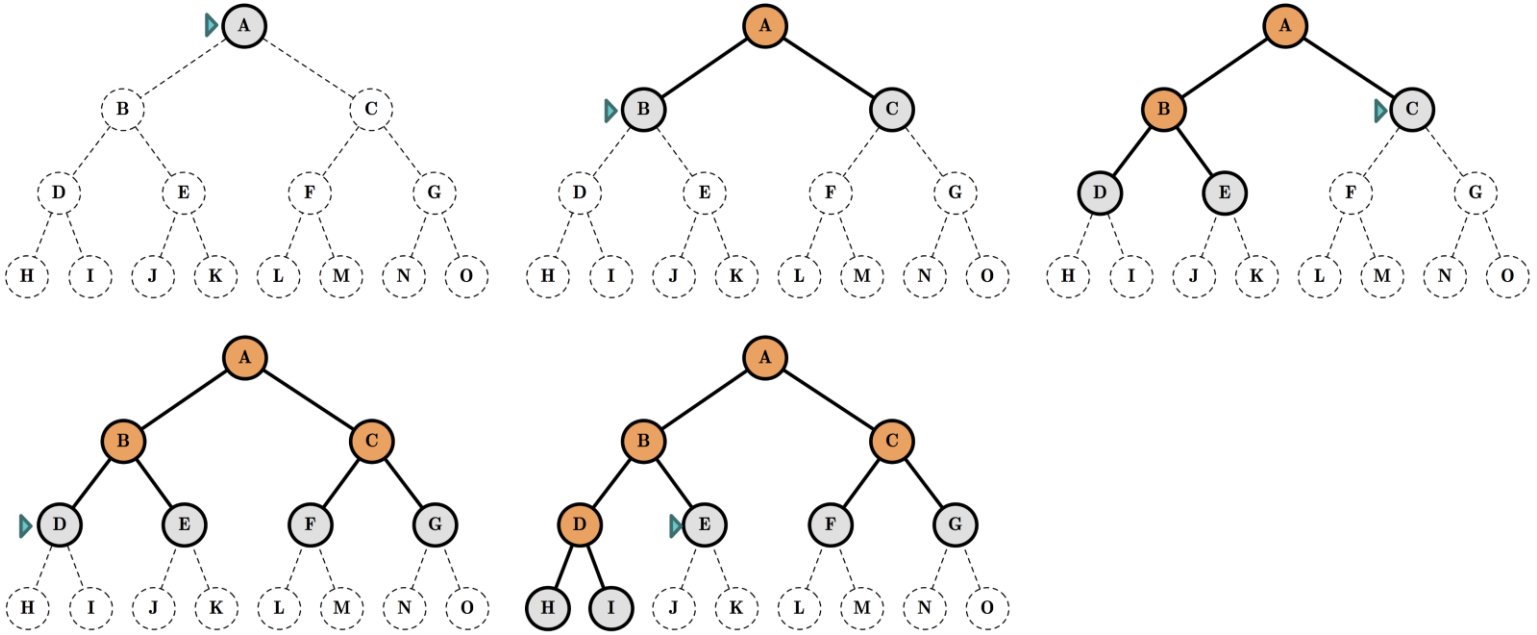
Lecture 1: Search
Breadth-First Search (BFS)

BFS:
Expand shallowest first.



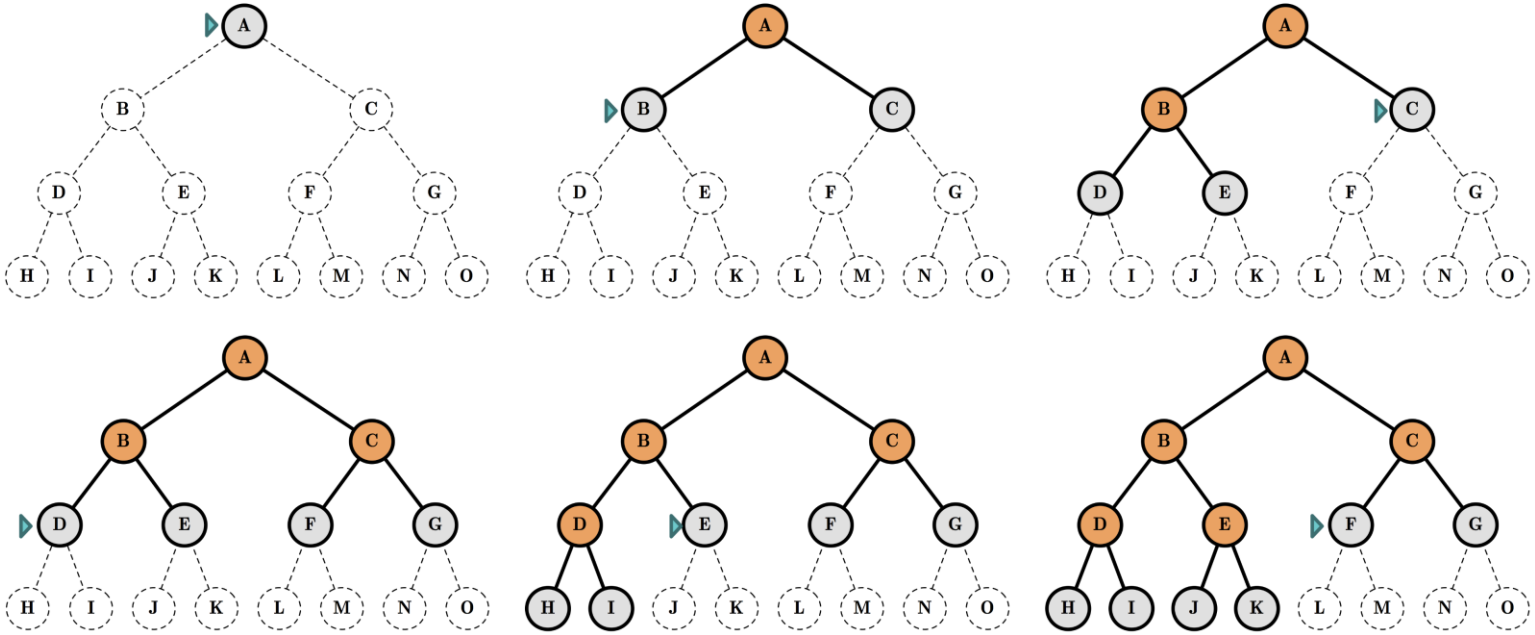
Lecture 1: Search
Breadth-First Search (BFS)

BFS:
Expand shallowest first.



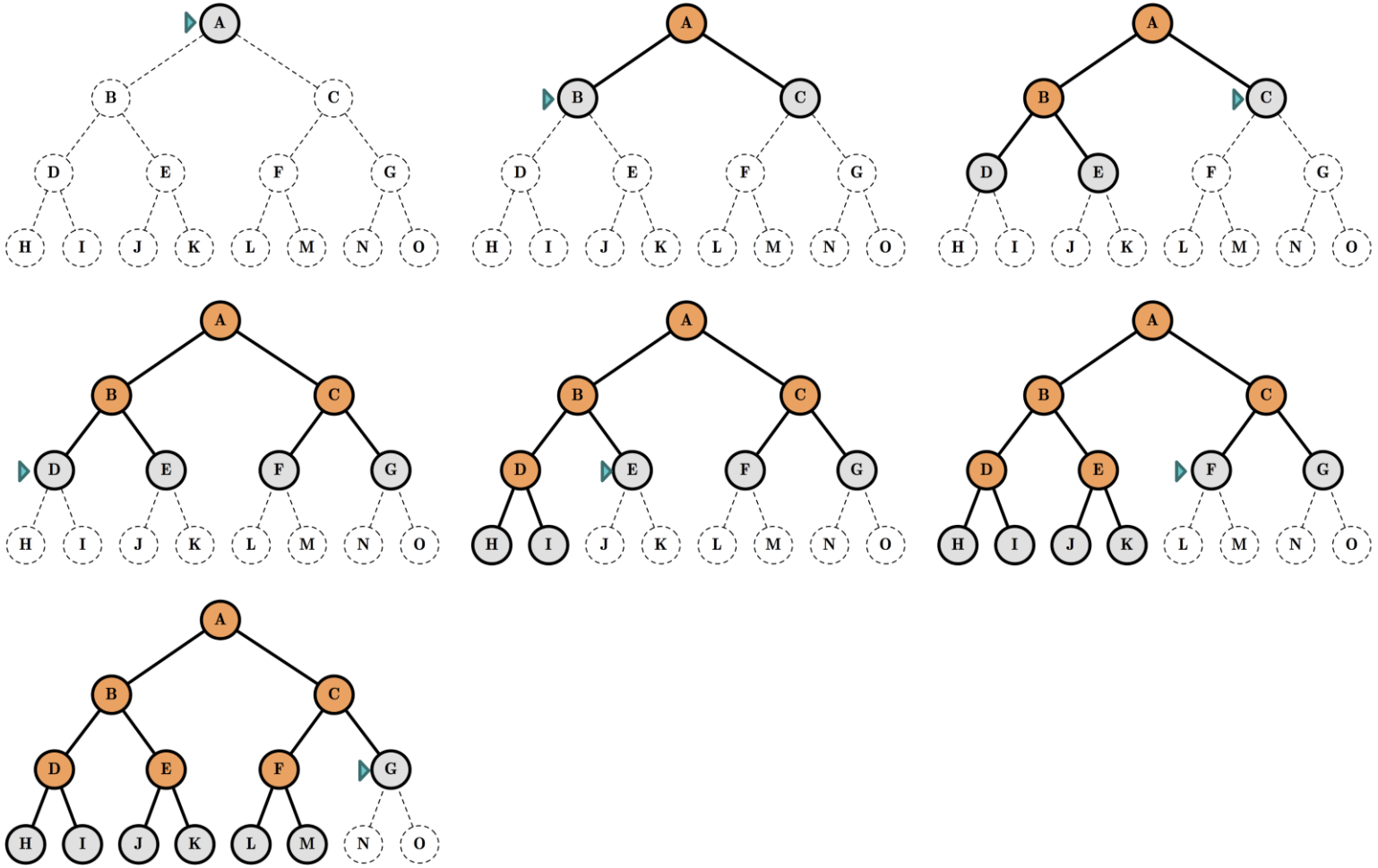
Lecture 1: Search
Breadth-First Search (BFS)

BFS:
Expand shallowest first.



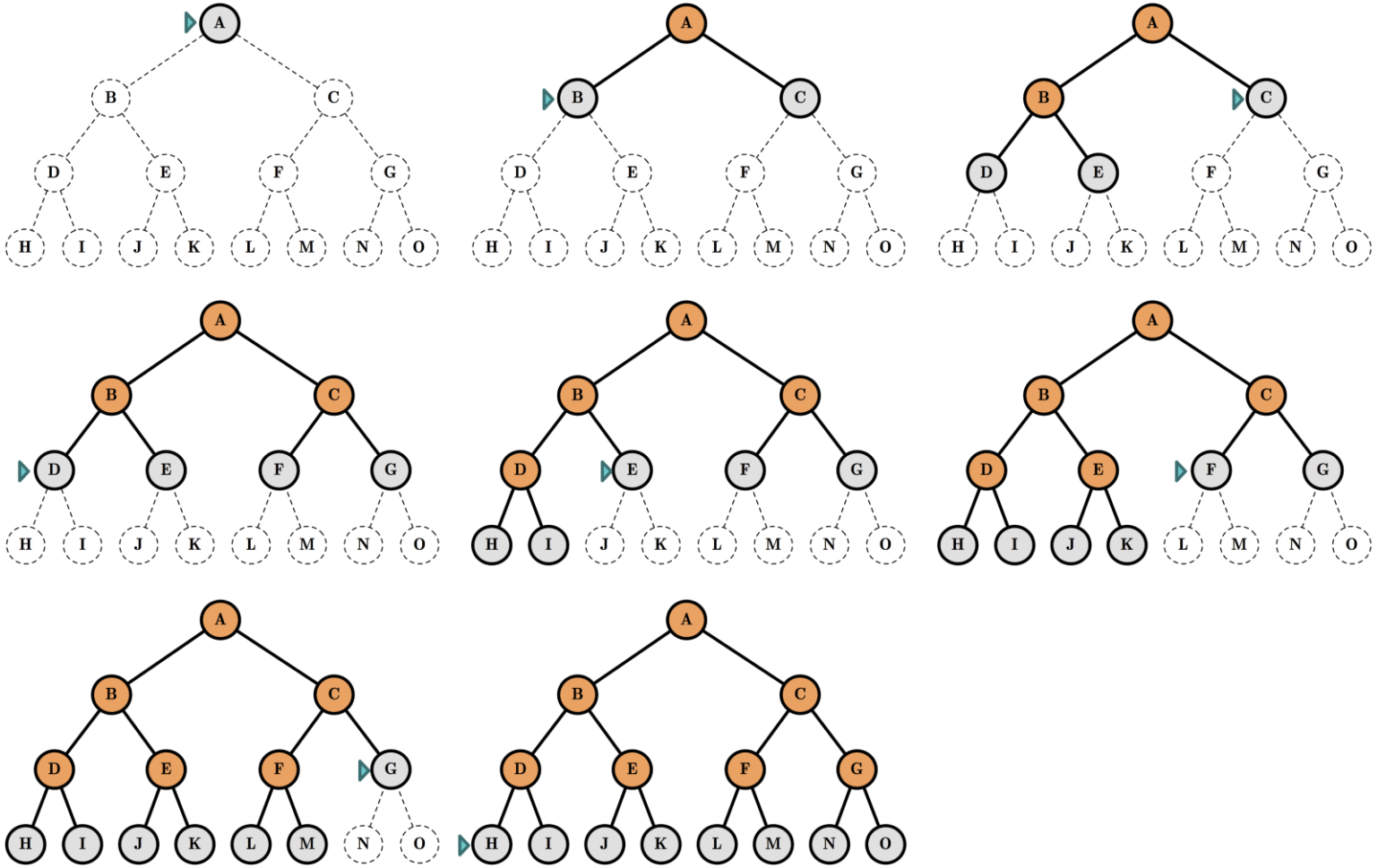
Lecture 1: Search
Breadth-First Search (BFS)

BFS:
Expand shallowest first.



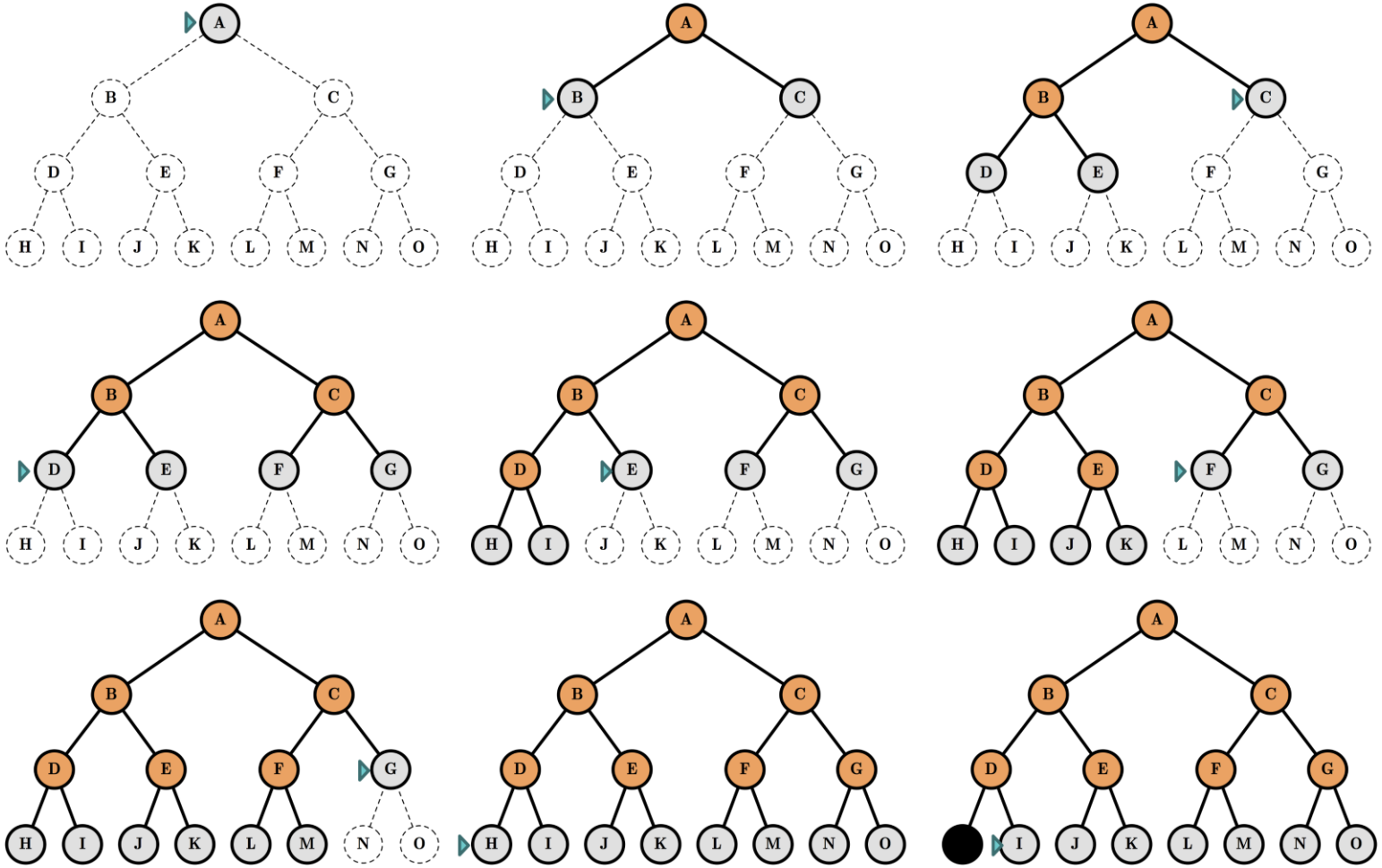
Lecture 1: Search
Breadth-First Search (BFS)

BFS:
Expand shallowest first.



Lecture 1: Search
Breadth-First Search (BFS)

BFS:
Expand shallowest first.



Lecture 1: Search
Breadth-First Search (BFS)/Criteria

- **Complete** Yes (if b is finite)
- **Time** $1 + b + b^2 + b^3 + \ldots + b^d = O(b^d)$
- **Space** $O(b^d)$
  Note: If the *goal test* is applied at expansion rather than generation then $O(b^{d+1})$
- **Optimal** Yes (if cost $= 1$ per step).
- **implementation**: fringe: FIFO (Queue)

Lecture 1: Search
Breadth-First Search (BFS)

Pros:

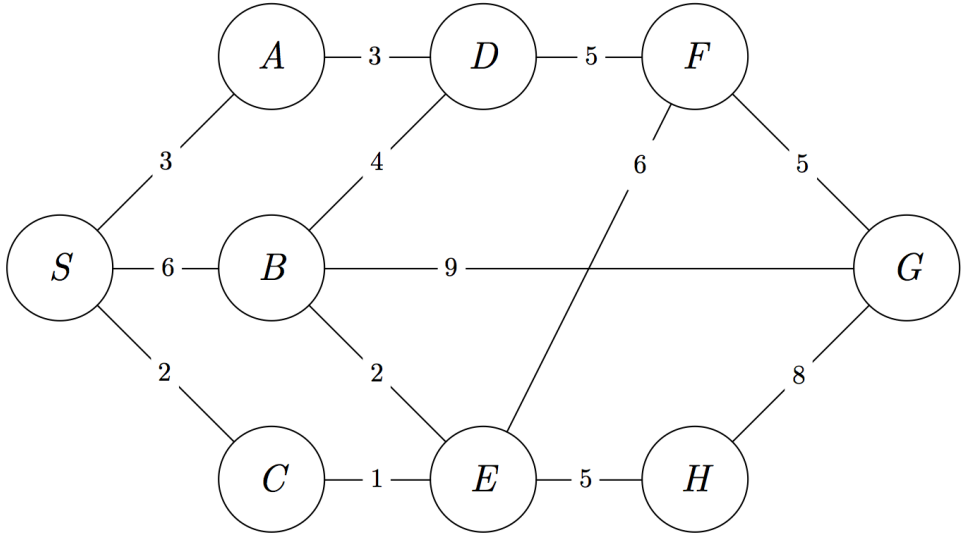
- This algorithm is guaranteed to find the optimal solution.

Cons:

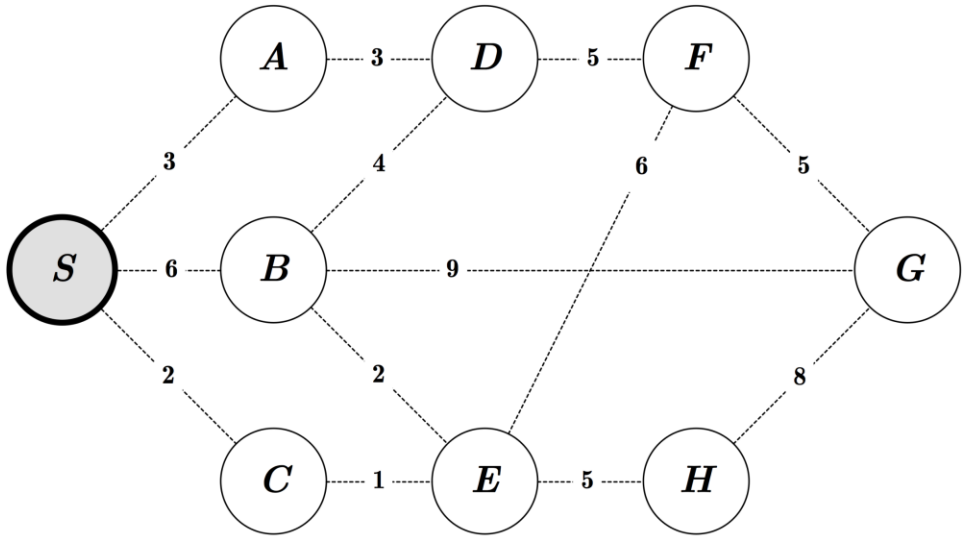
- This algorithm is almost guaranteed to take longer than the minimal time to run.
- At worst, this algorithm takes the longest possible time to run

Lecture 1: Search
Breadth-First Search (BFS)/Example 2

BFS:
Expand shallowest first.
Checked



Lecture 1: Search
Breadth-First Search (BFS)/Example 2

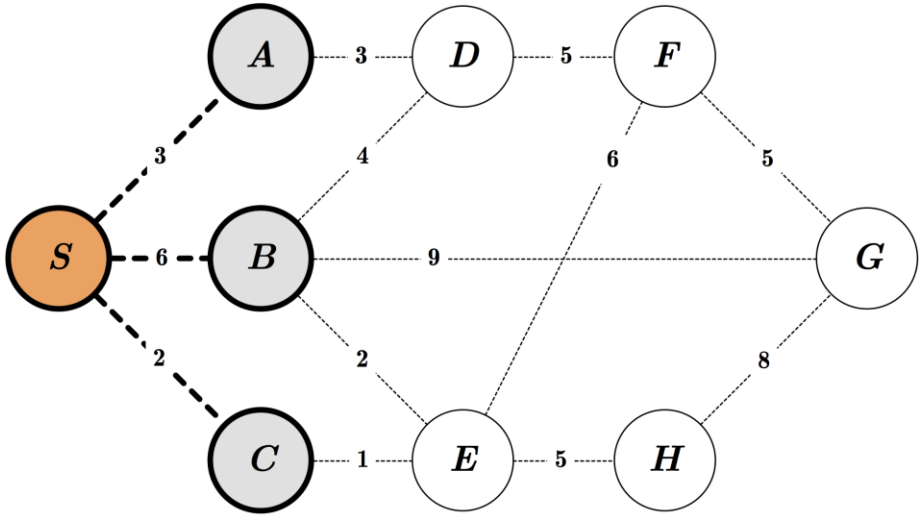


Queue:

$\boxed{S}$

Order of Visit:

Lecture 1: Search
Breadth-First Search (BFS)/Example 2



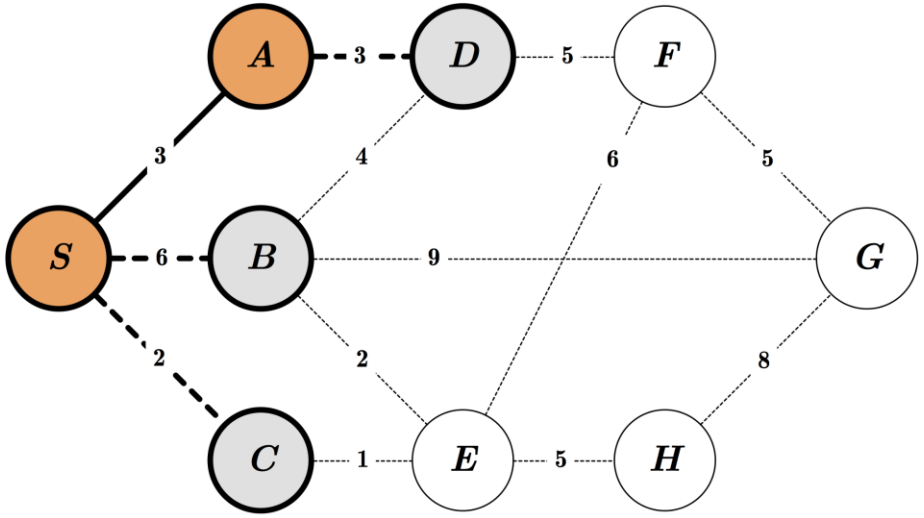
Queue:

S	*A*	*B*	*C*

Order of Visit:

S

Lecture 1: Search
Breadth-First Search (BFS)/Example 2



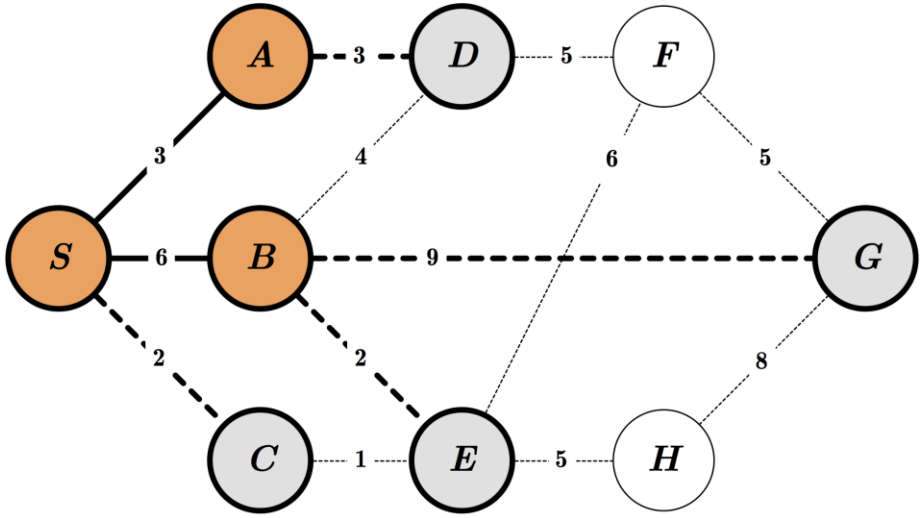
Queue:

S	*A*	*B*	*C*	*D*

Order of Visit:

S *A*

Lecture 1: Search
Breadth-First Search (BFS)/Example 2



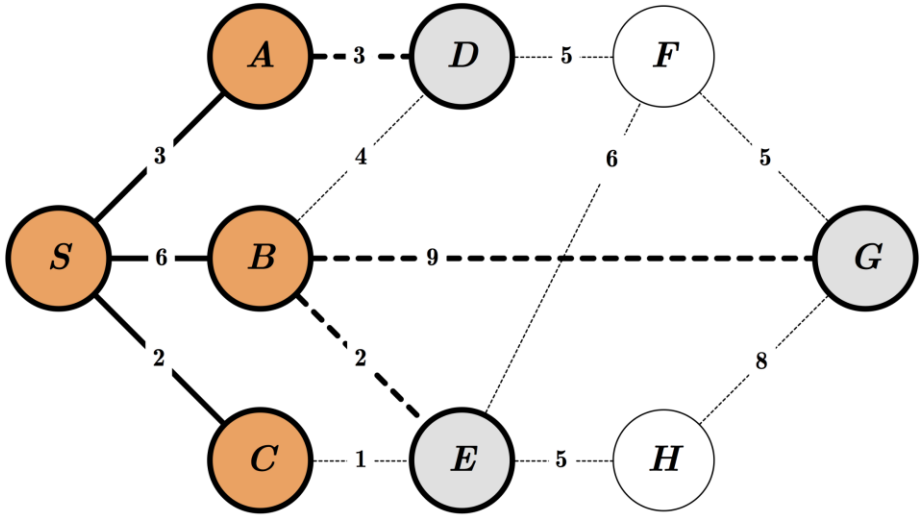
Queue:

S	*A*	*B*	*C*	*D*	*E*	*G*

Order of Visit:

S *A* *B*

Lecture 1: Search
Breadth-First Search (BFS)/Example 2



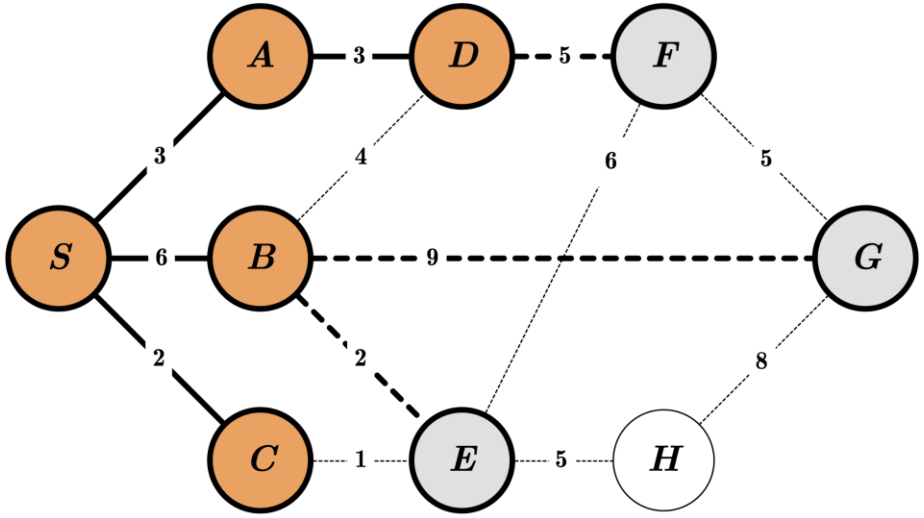
Queue:

S	*A*	*B*	*C*	*D*	*E*	*G*

Order of Visit:

S *A* *B* *C*

Lecture 1: Search
Breadth-First Search (BFS)/Example 2



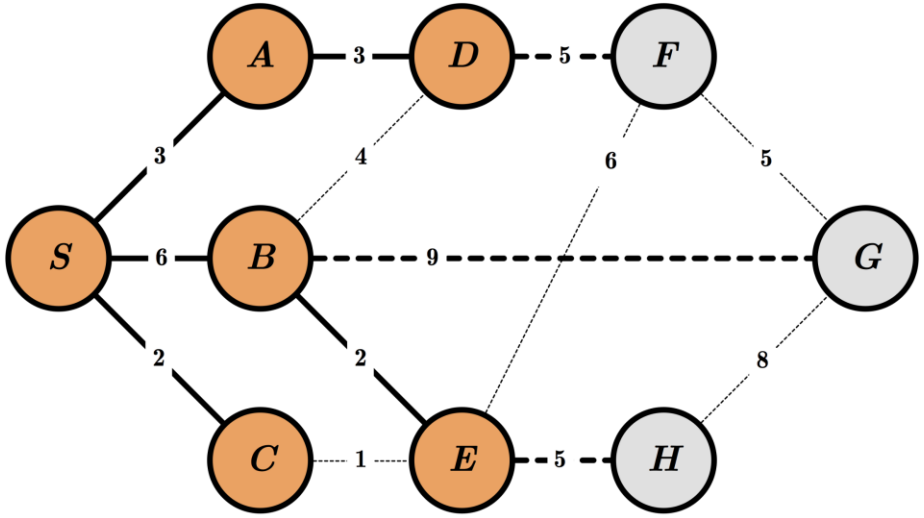
Queue:

S	*A*	*B*	*C*	*D*	*E*	*G*	*F*

Order of Visit:

S *A* *B* *C* *D*

Lecture 1: Search
Breadth-First Search (BFS)/Example 2



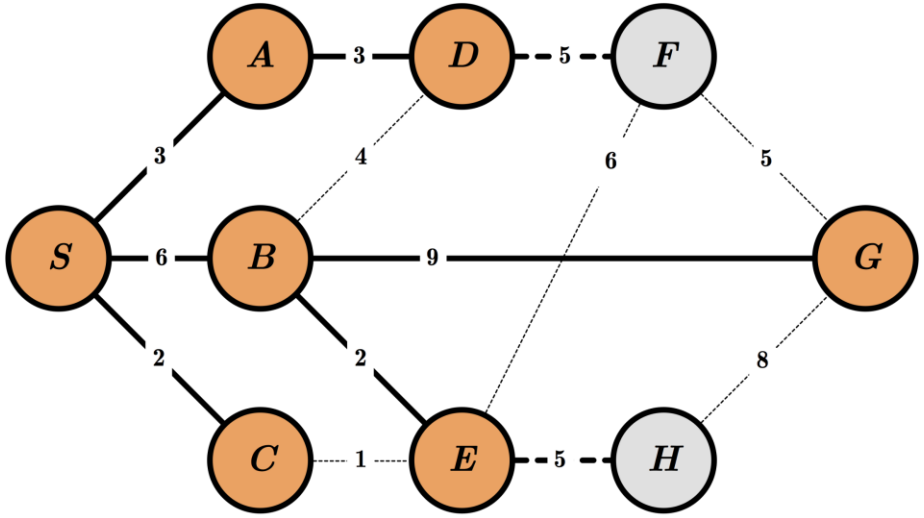
Queue:

S	*A*	*B*	*C*	*D*	*E*	*G*	*F*	*H*

Order of Visit:

S *A* *B* *C* *D* *E*

Lecture 1: Search
Breadth-First Search (BFS)/Example 2



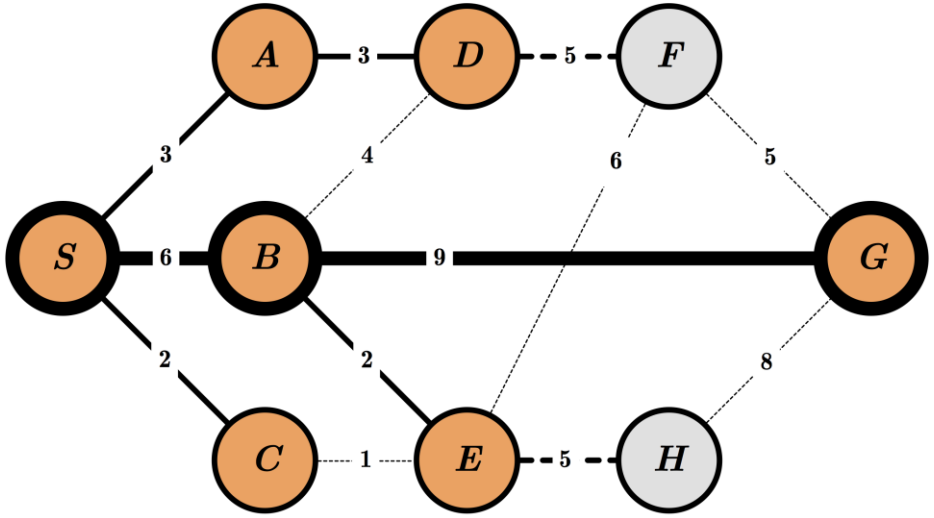
Queue:

S	*A*	*B*	*C*	*D*	*E*	*G*	*F*	*H*

Order of Visit:

S *A* *B* *C* *D* *E* *G*

Lecture 1: Search
Breadth-First Search (BFS)/Example 2



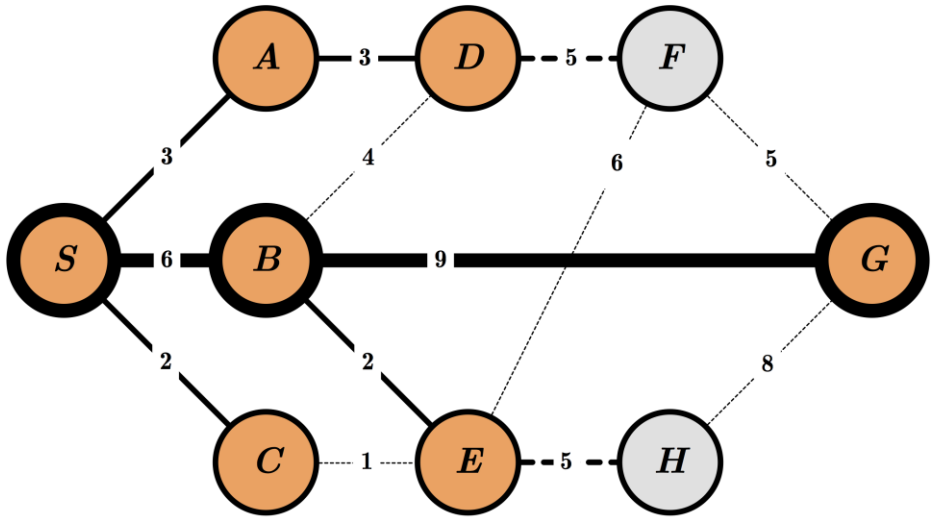
Queue:

S	*A*	*B*	*C*	*D*	*E*	*G*	*F*	*H*

Order of Visit:

S ***A*** ***B*** ***C*** ***D*** ***E*** ***G***

Lecture 1: Search
BFS VS DFS

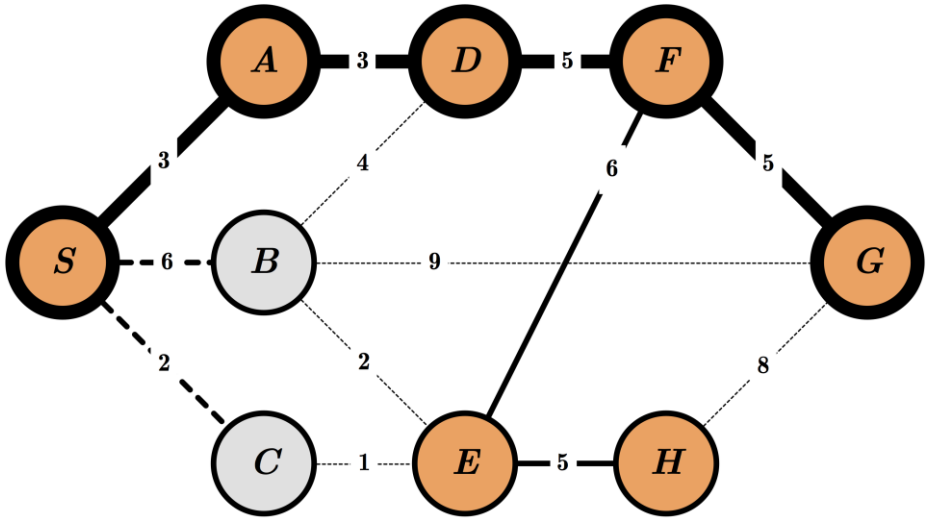


Queue:

S	*A*	*B*	*C*	*D*	*E*	*G*	*F*	*H*

Order of Visit:

S *A* *B* *C* *D* *E* *G*



Stack:

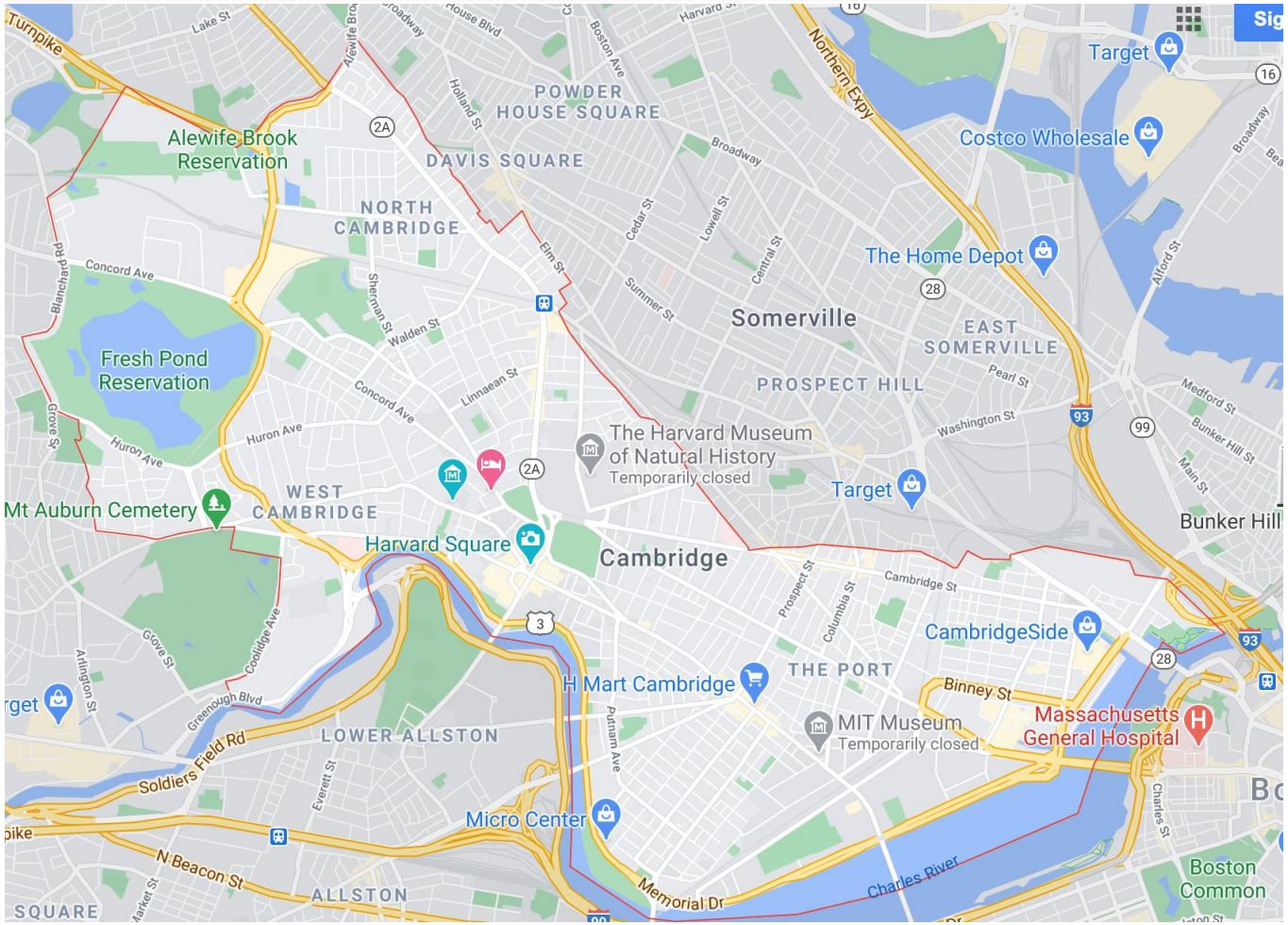
S	*C*	*B*	*A*	*D*	*F*	*G*	*E*	*H*

Order of Visit:

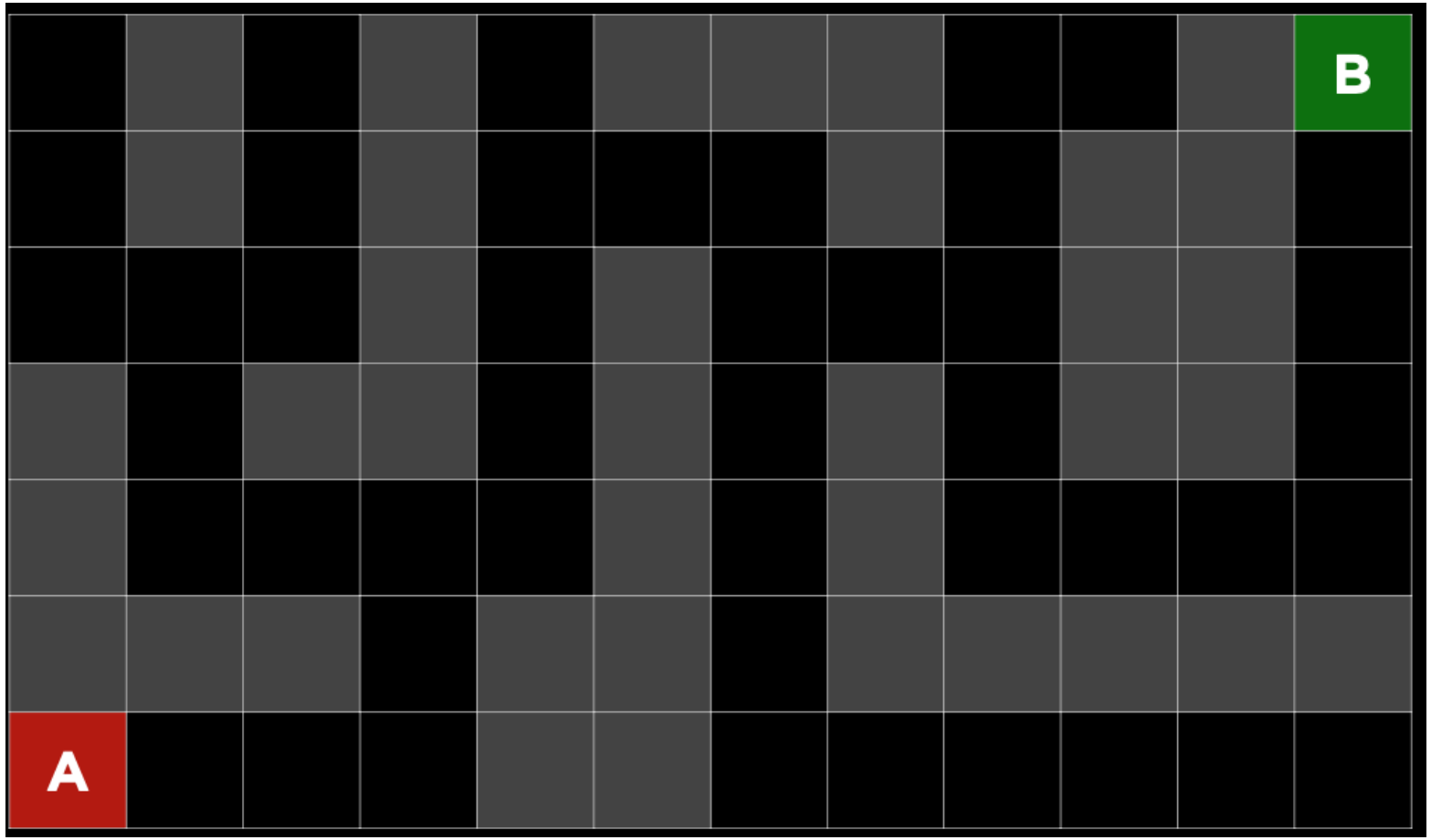
S *A* *D* *F* *E* *H* *G*

Lecture 1: Search
BFS VS DFS

Demo

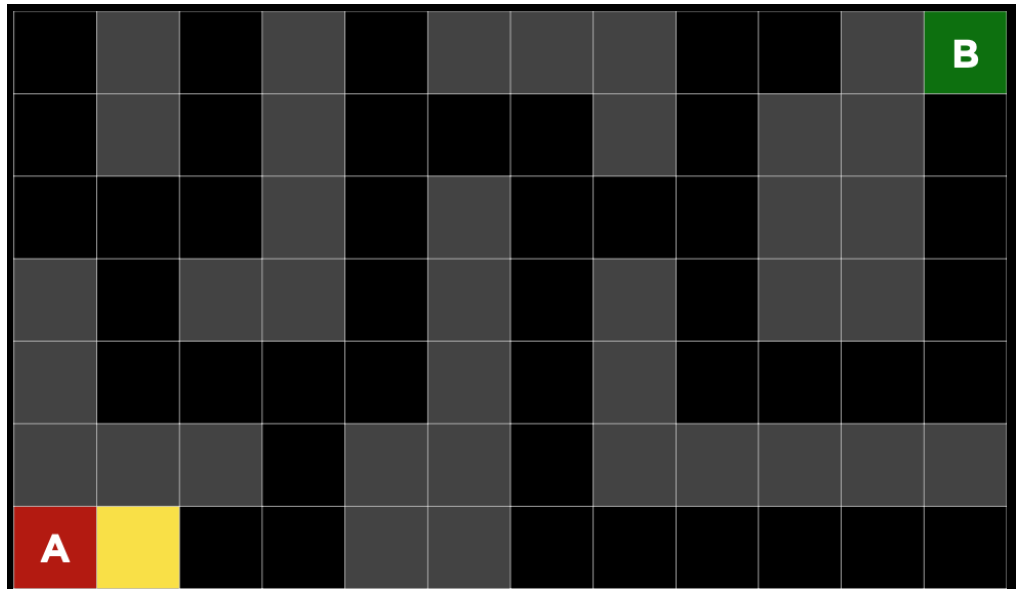


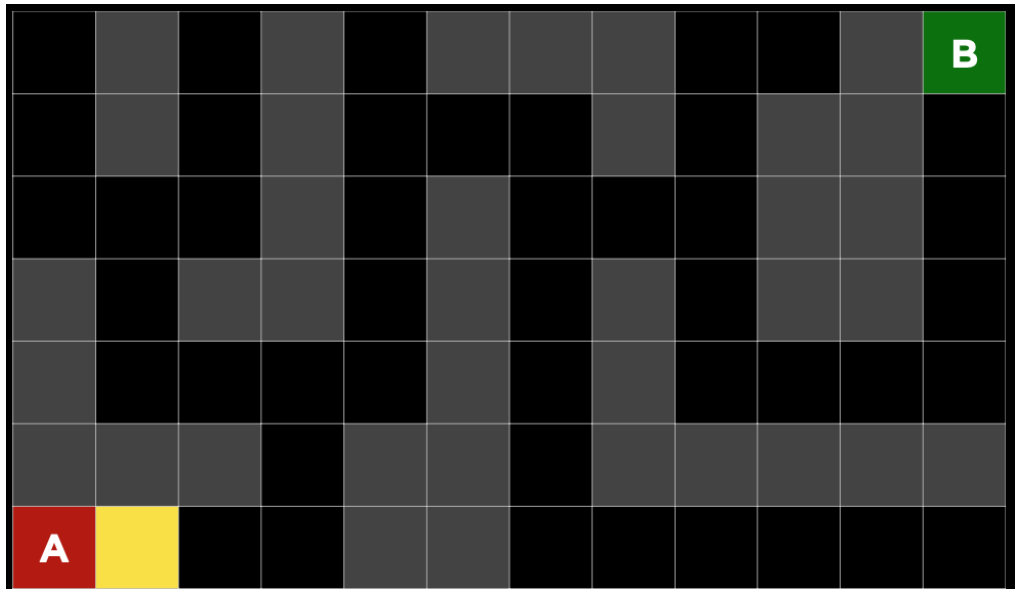
Lecture 1: Search
DFS VS BFS



Lecture 1: Search

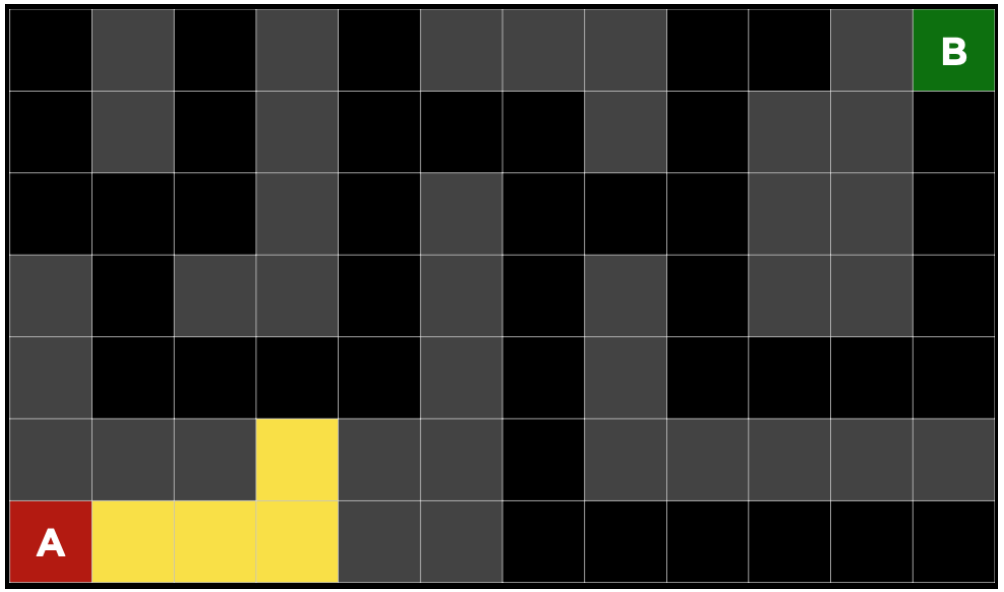
DFS VS BFS

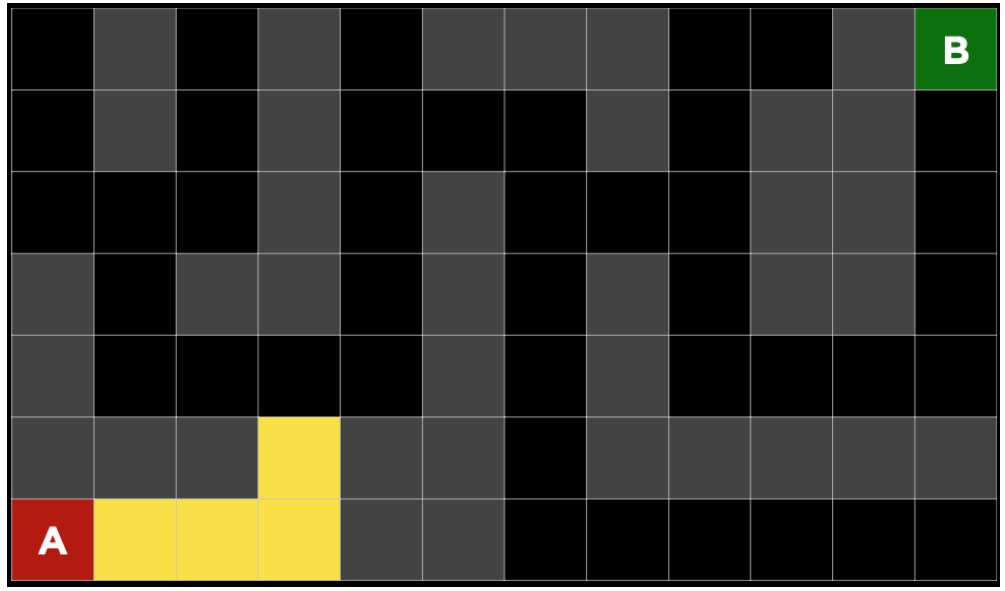




Lecture 1: Search

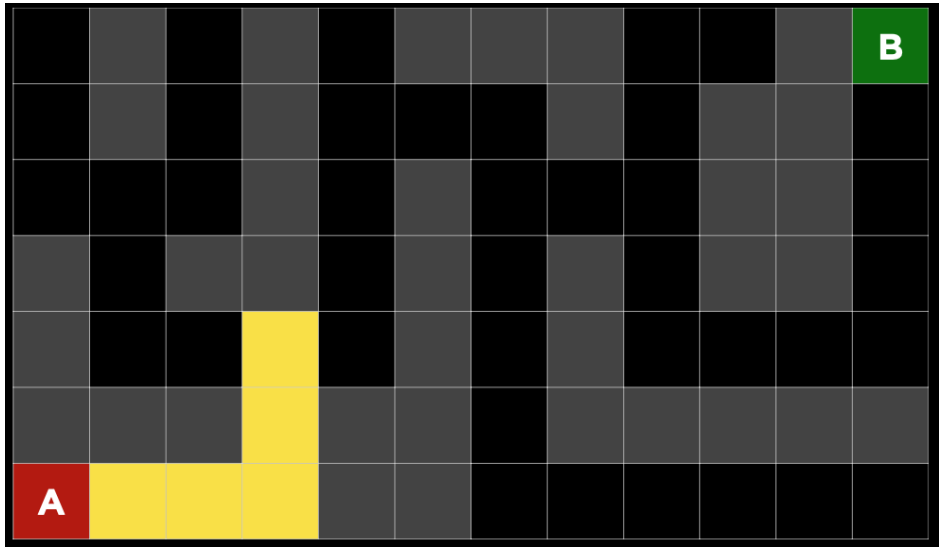
DFS VS BFS

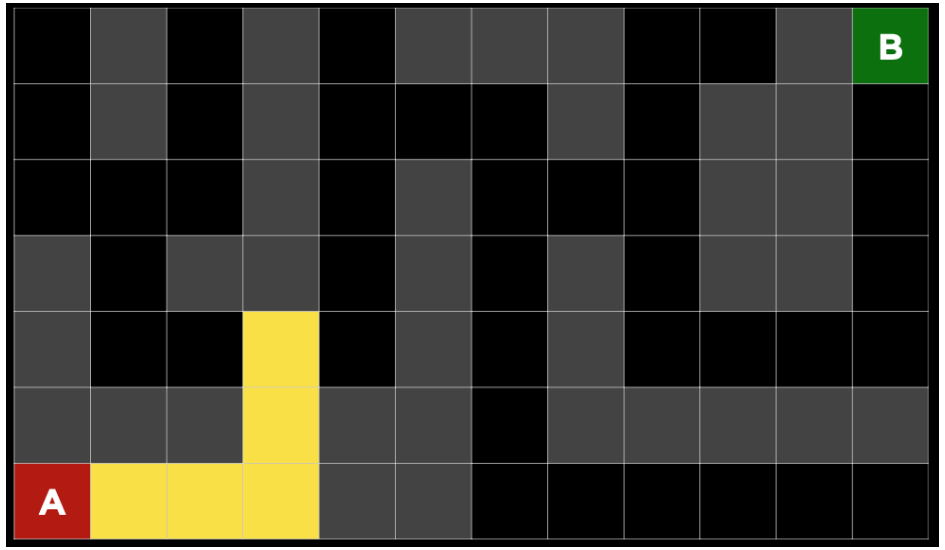




Lecture 1: Search

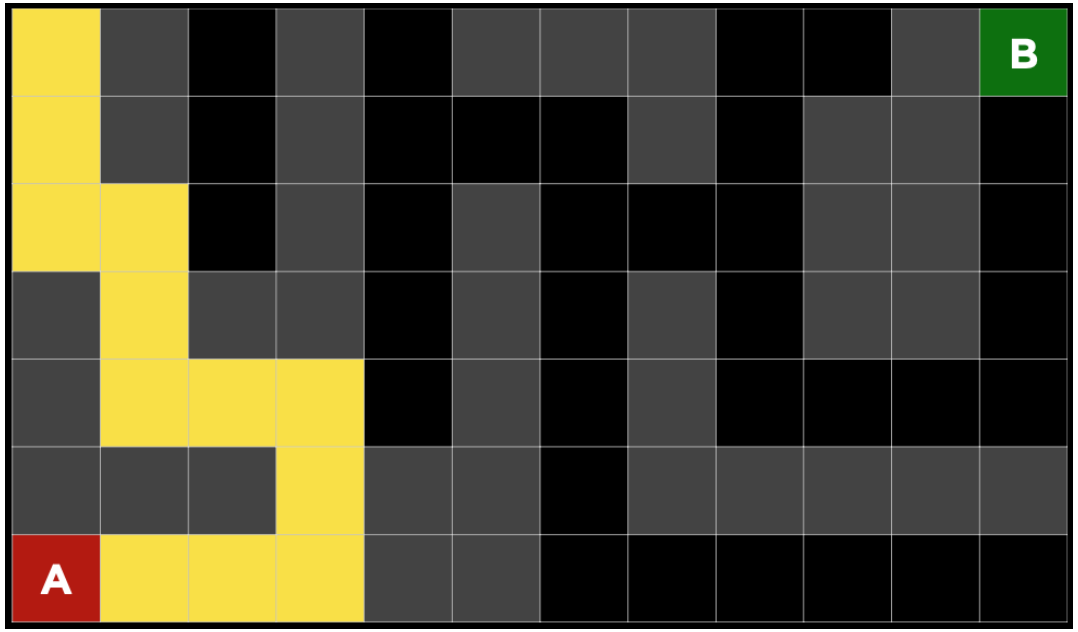
DFS VS BFS

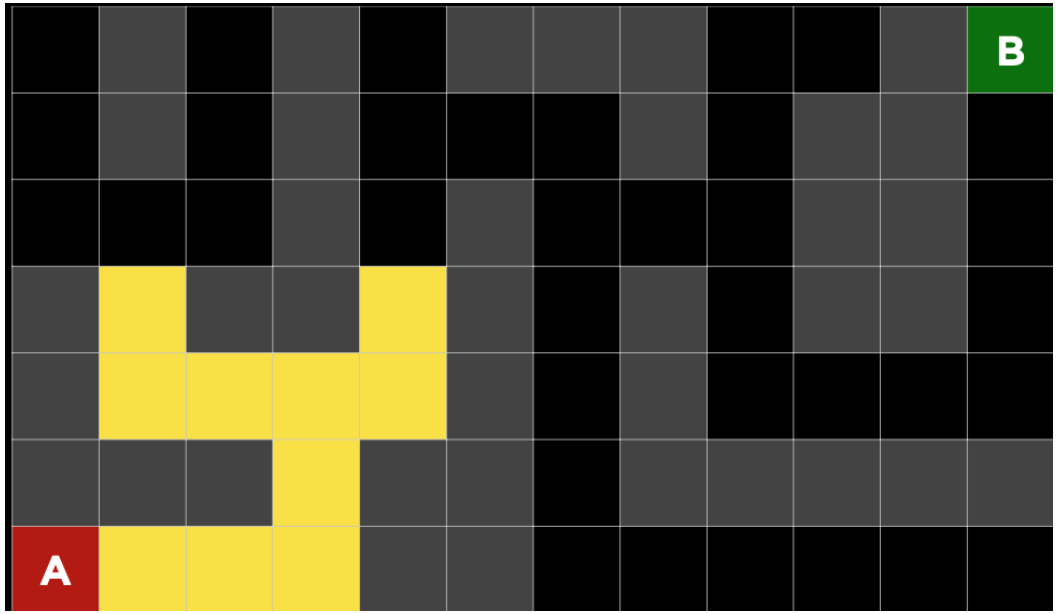




Lecture 1: Search

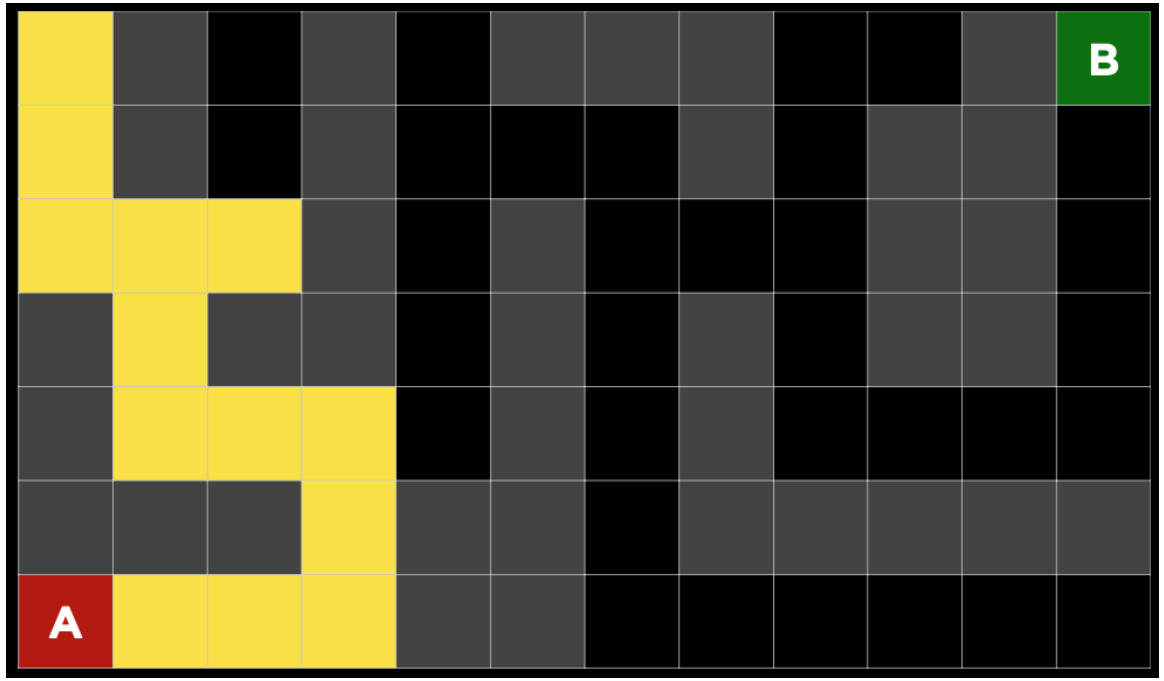
DFS VS BFS

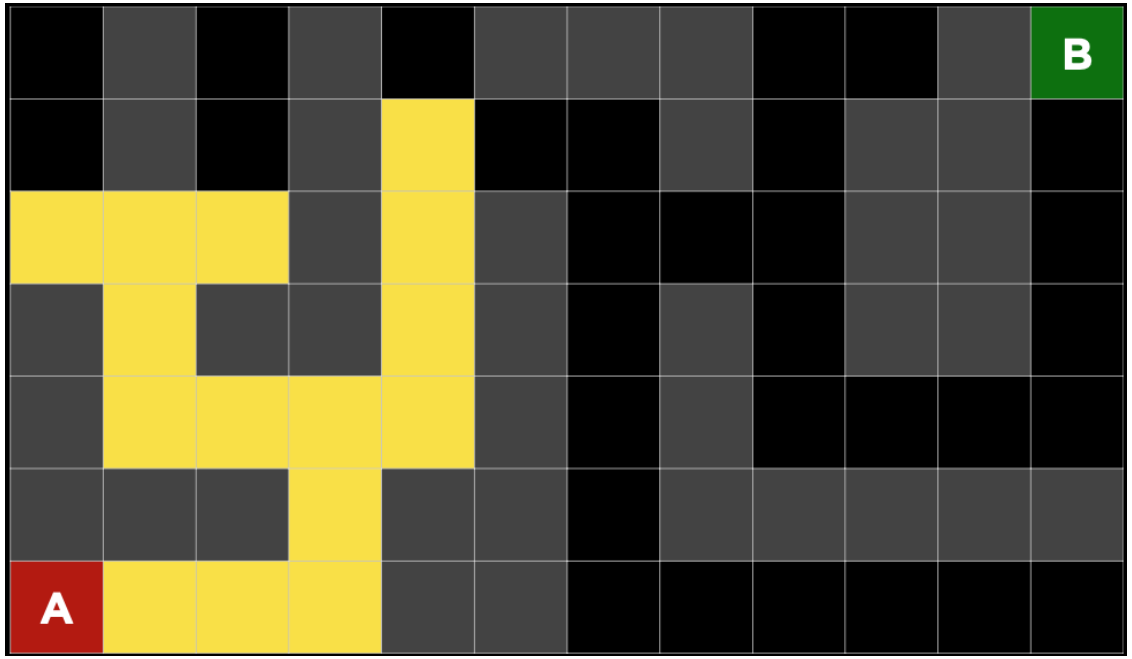




Lecture 1: Search

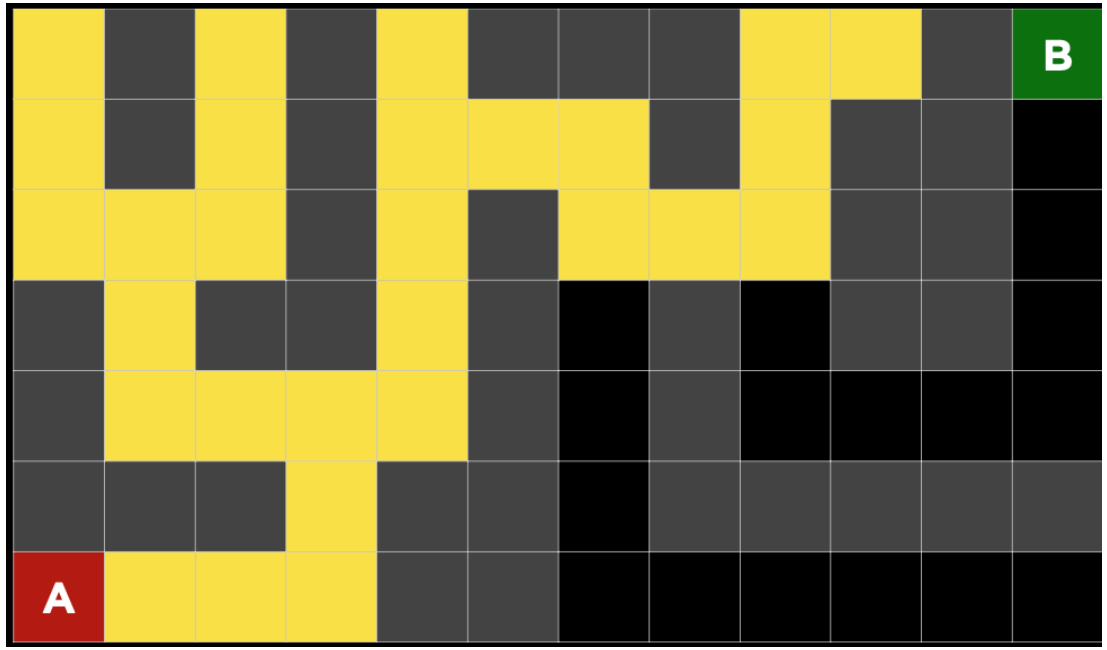
DFS VS BFS

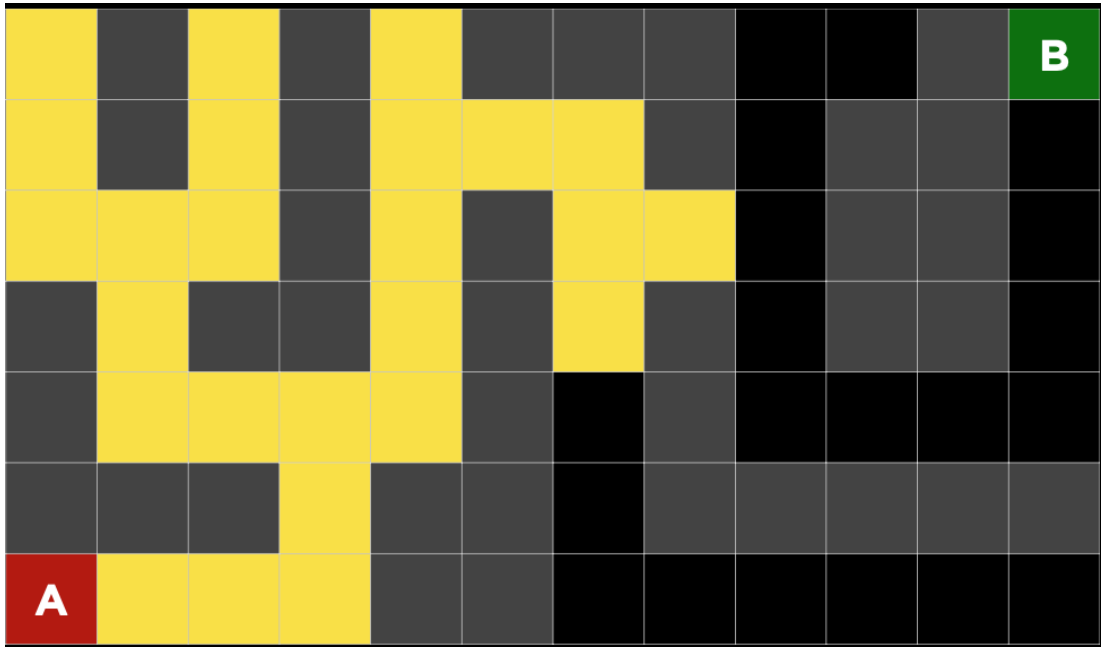




Lecture 1: Search

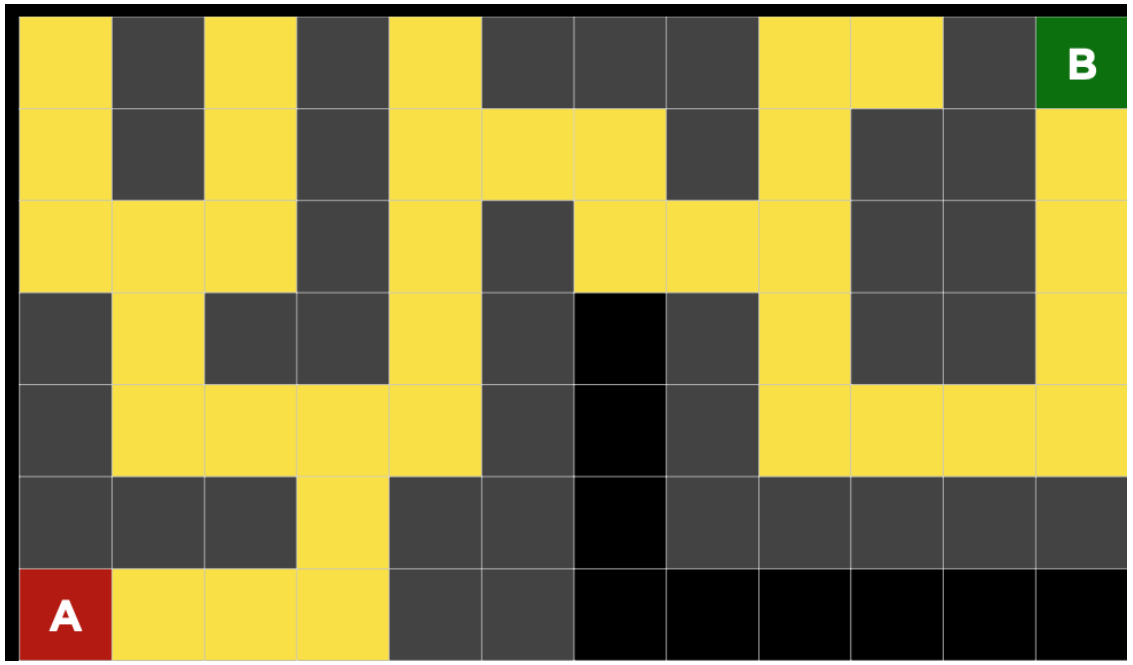
DFS VS BFS

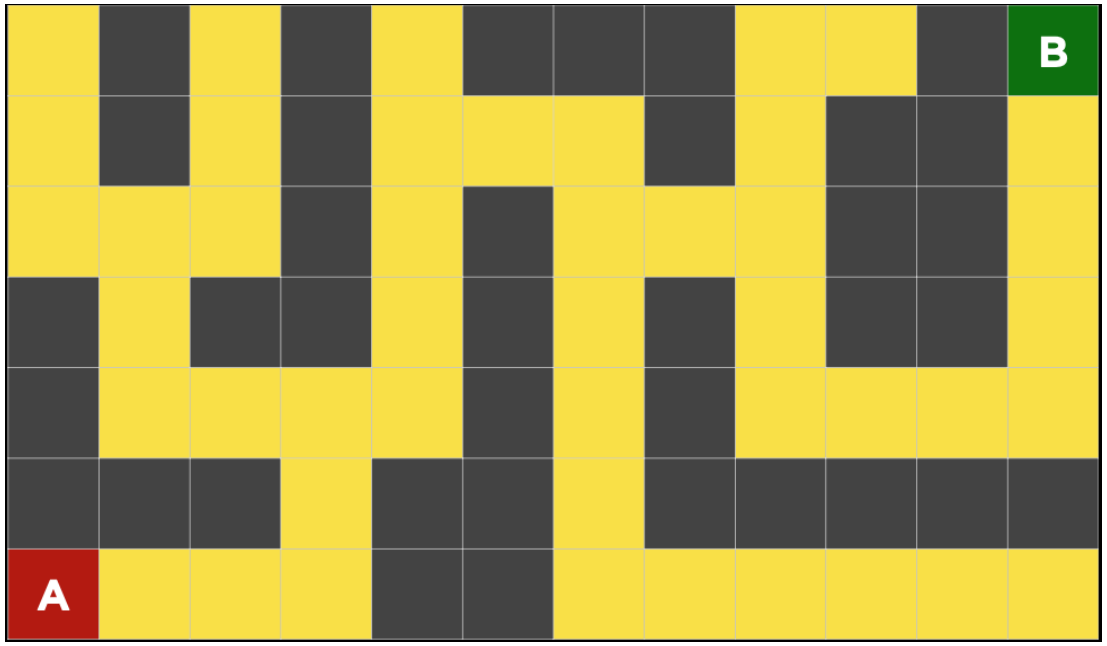




Lecture 1: Search

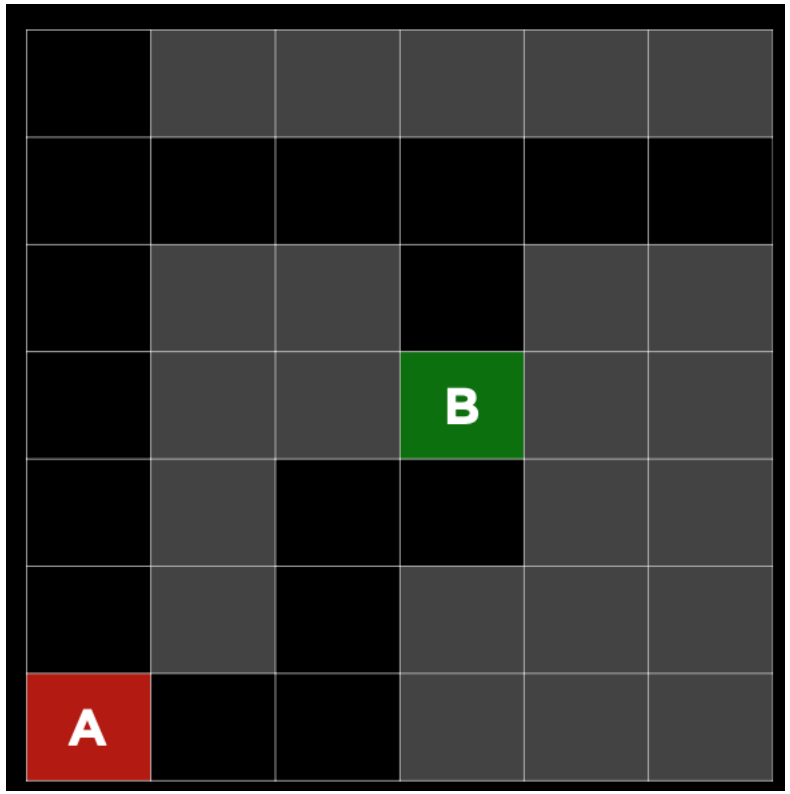
DFS VS BFS

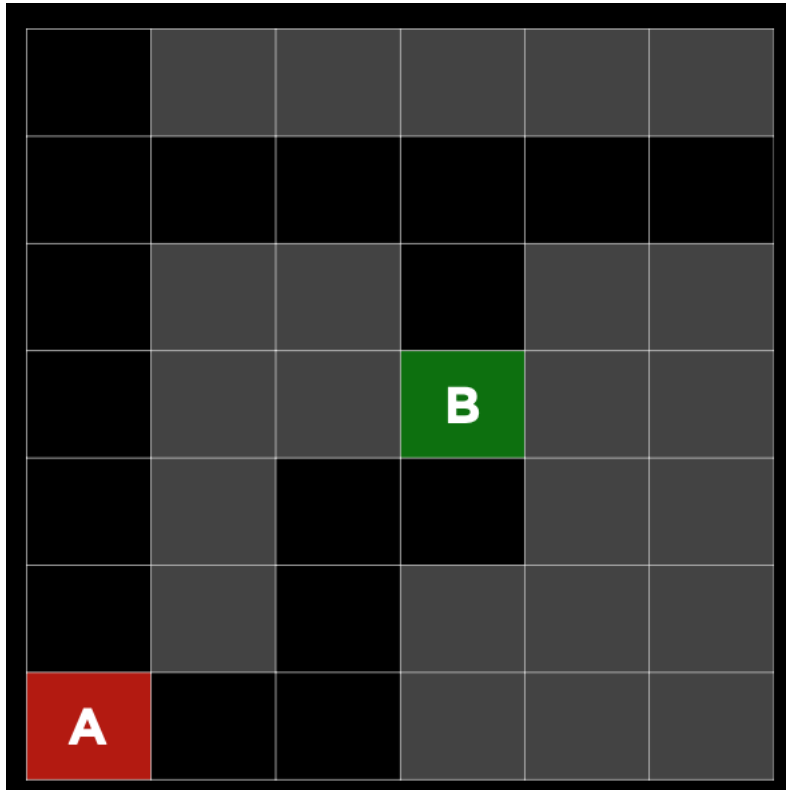




Lecture 1: Search

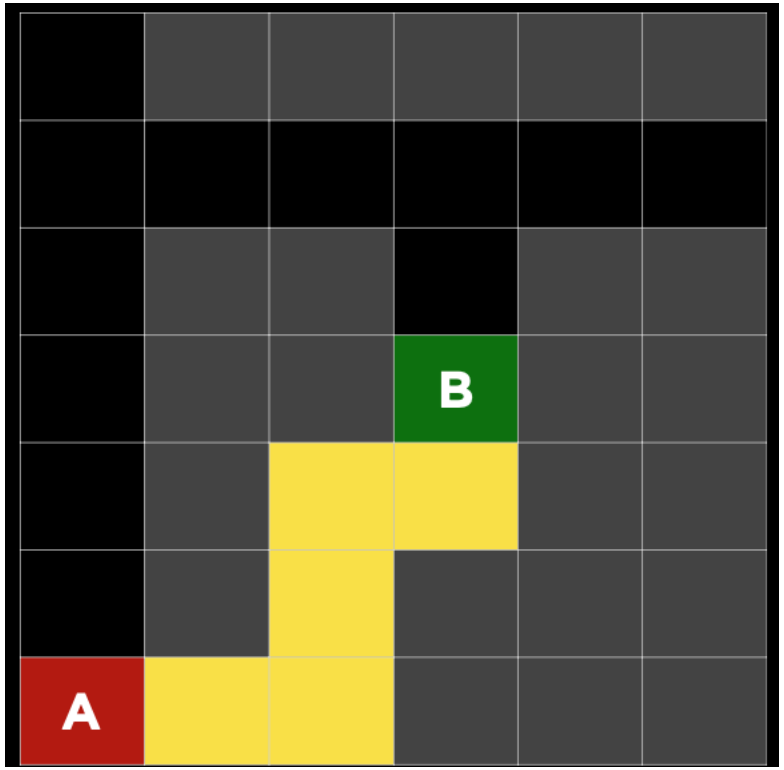
DFS VS BFS

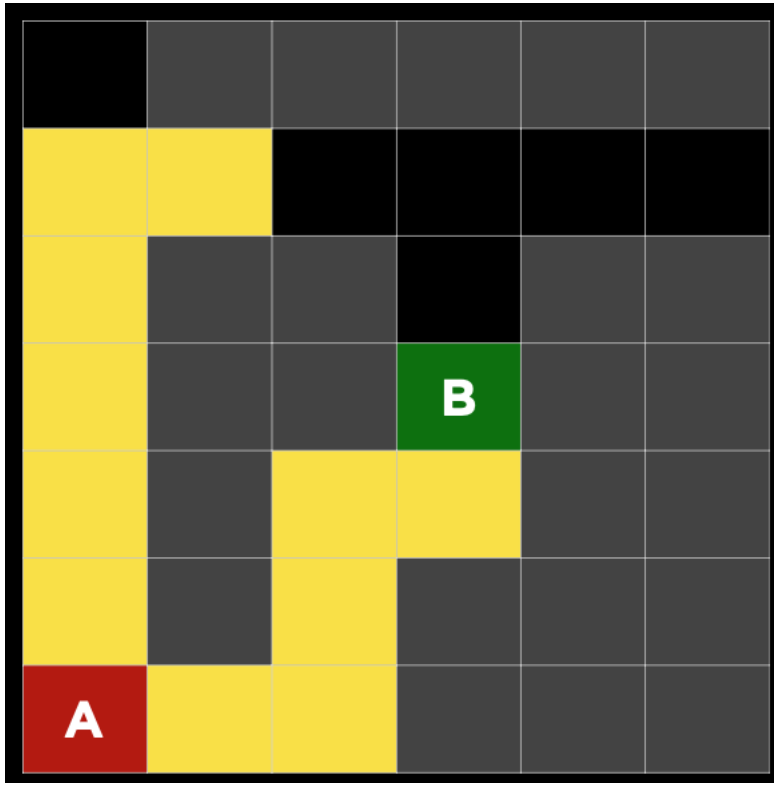




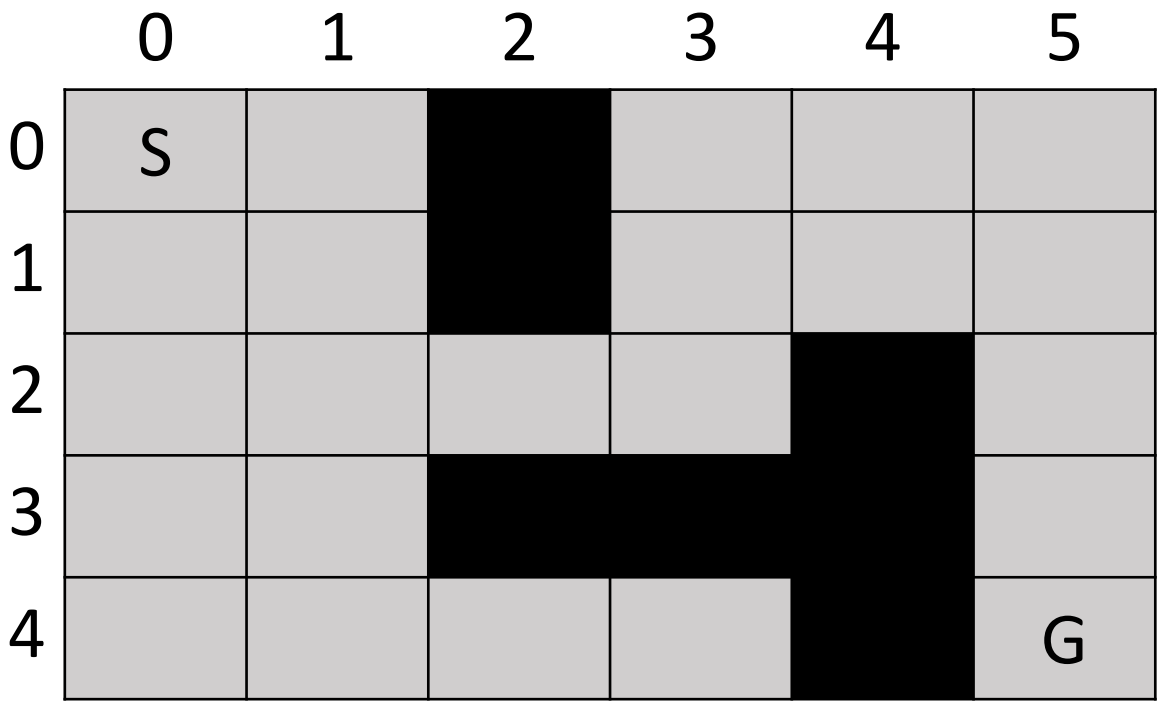
Lecture 1: Search

DFS VS BFS





Lecture 1: Search
BFS and DFS Exercise



Lecture 1: Search informed search

Search strategy that uses problem-specific knowledge to find solutions more efficiently

- Best-First Search
  - Greedy Best-First Search
  - A* (*Robot motion planning*)
  - Beam Search (*Machine translation*)

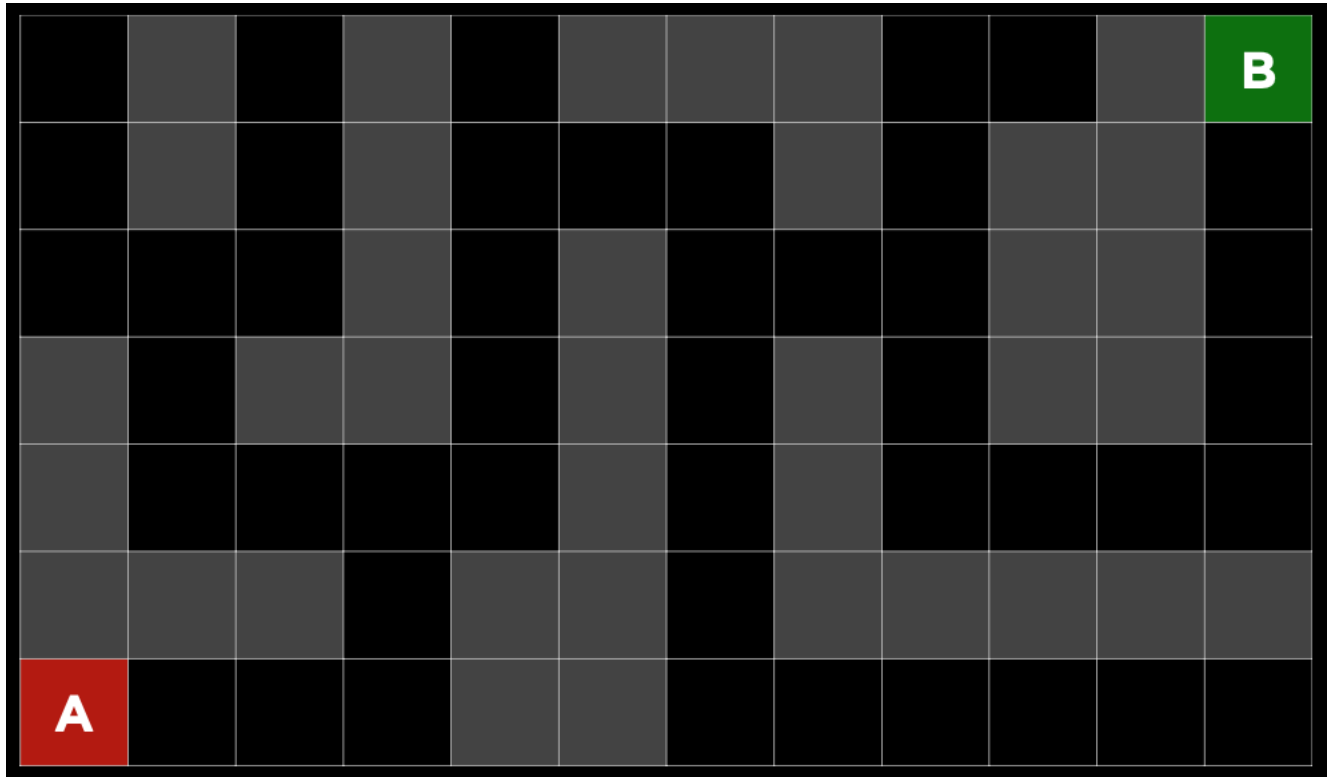
Lecture 1: Search
Greedy Best-First Search

Search algorithm that expands the node that is closest to the goal,

as estimated by a heuristic function h(n)

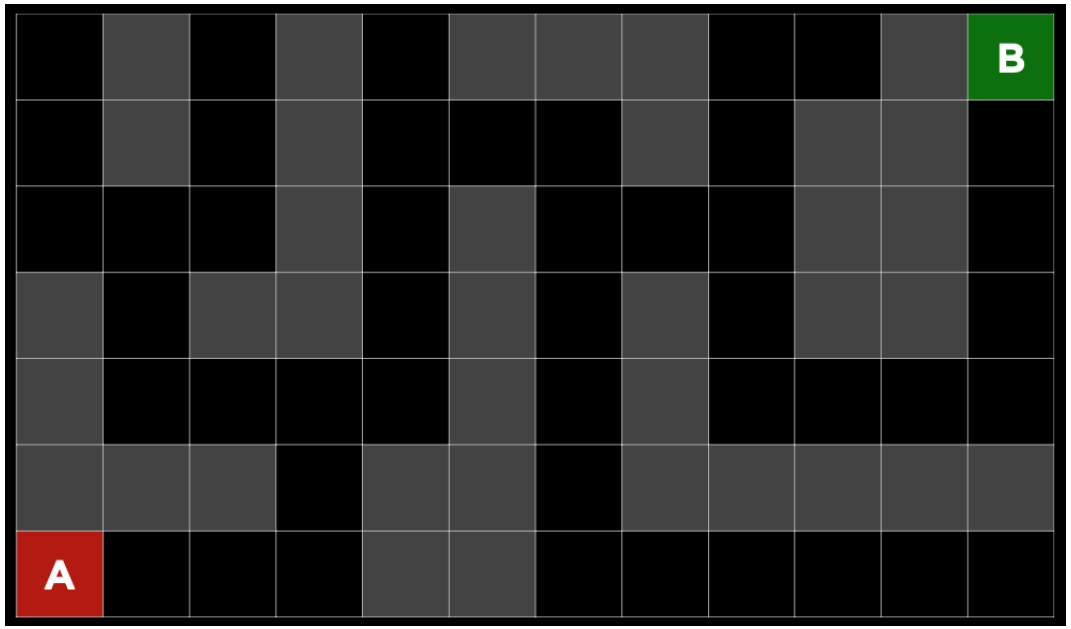
Manhattan distance

Lecture 1: Search
Greedy Best-First Search



Lecture 1: Search
Greedy Best-First Search

Heuristic function?



11	10	9	8	7	6	5	4	3	2	1	0
										2	1
											2
											3
											4
											5
	16										6

Lecture 1: Search
Greedy Best-First Search

Heuristic function?

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

11	10	9	8	7	6	5	4	3	2	1	0
										2	1
											2
											3
											4
											5
	16										6

Lecture 1: Search
Greedy Best-First Search

Heuristic function?

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

Lecture 1: Search
Greedy Best-First Search

- Heuristic function?

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

Lecture 1: Search
Greedy Best-First Search

- Heuristic function?

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

Lecture 1: Search
Greedy Best-First Search

- Heuristic function?

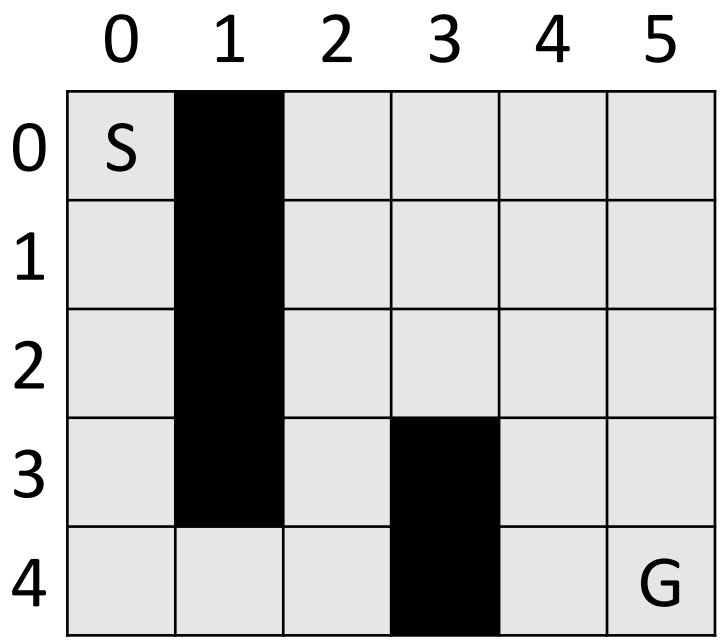
11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

Lecture 1: Search
Greedy Best-First Search

- Heuristic function?

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

Lecture 1: Search Exercise



	0	1	2	3	4	5
0	9	8	7	6	5	4
1	8	7	6	5	4	3
2	7	6	5	4	3	2
3	6	5	4	3	2	1
4	5	4	3	2	1	0

$$h(x, y) \leq \text{ distance to goal from } x, y$$

Lecture 1: Search
A* Search

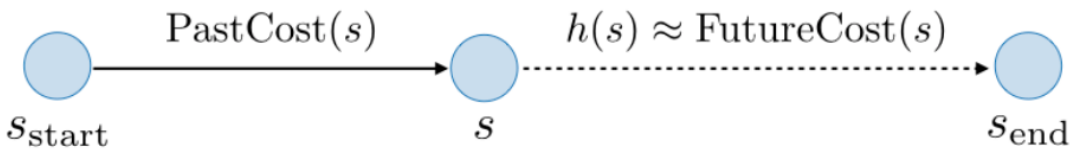
Search algorithm that expands node with lowest value of

$f(n) = g(n) + h(n)$

- $g(n)$ = cost to reach node
- $h(n)$ = estimated cost to goal

Lecture 1: Search
A* Search

- **Heuristic function** — A heuristic is a function h over states s, where each $h(s)$ aims at estimating $\text{FutureCost}(s)$, the cost of the path from s to s_{end}.



Lecture 1: Search
A* Search

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	16	15	14		12	11	10	9	8	7	6

$$f = g + h(x, y)$$

Lecture 1: Search
A* Search

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	1+16	15	14		12	11	10	9	8	7	6

$$f = g + h(x, y)$$

Lecture 1: Search
A* Search

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

$$f = g + h(x, y)$$

Lecture 1: Search
A* Search

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		10	9	8	7	6	5	4		2
	13		6+11						5		3
	14	13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

$$f = g + h(x, y)$$

Lecture 1: Search
A* Search

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	13		6+11						14+5		3
	14	13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

$$f = g + h(x, y)$$

Lecture 1: Search
A* Search

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	13		6+11						14+5		3
	14	6+13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

$$f = g + h(x, y)$$

Lecture 1: Search
A* Search

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	13		6+11						14+5		3
	14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

$$f = g + h(x, y)$$

Lecture 1: Search
A* Search

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

$$f = g + h(x, y)$$

Lecture 1: Search
A* Search

	10	9	8	7	6	5	4	3	2	1	B
	11										1
	12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

$$f = g + h(x, y)$$

Lecture 1: Search
A* Search

	11+10	12+9	13+8	14+7	15+6	16+5	17+4	18+3	19+2	20+1	B
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

$$f = g + h(x, y)$$

Lecture 1: Search
A* Search

DARPA Grand Challenge

https://www.youtube.com/watch?v

Lecture 1: Search Exercise

	0	1	2	3	4	5
0	S					
1						
2						
3						
4						G

	0	1	2	3	4	5
0	9	8	7	6	5	4
1	8	7	6	5	4	3
2	7	6	5	4	3	2
3	6	5	4	3	2	1
4	5	4	3	2	1	0

$$f = g + h(x, y)$$

$$OPEN\ [x, y]\ g, f$$

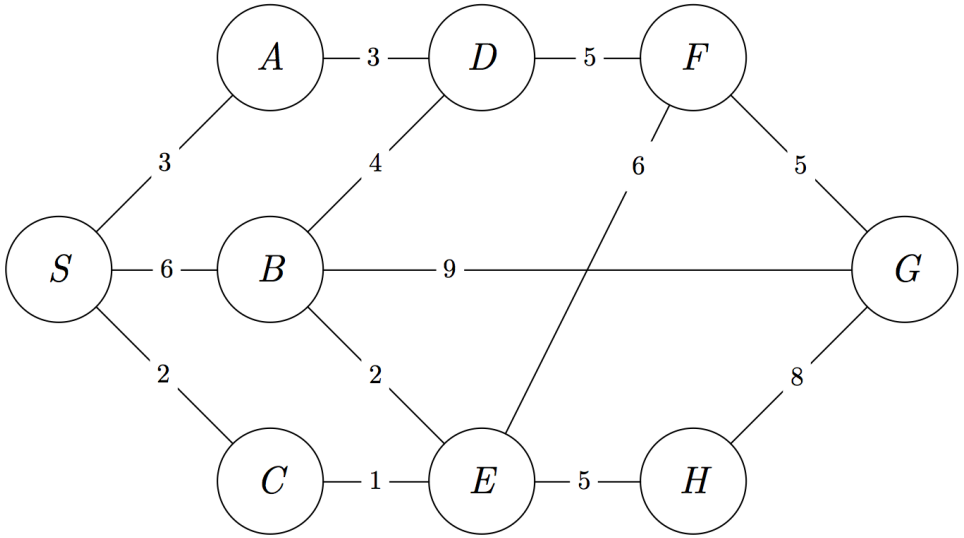
$$[0,0]\ 0, 9$$

$$[4,1]5, 9 \rightarrow [4,2]6,9 \rightarrow [3,2]7,11\ or\ [4,3]7,9$$

$$h(x, y)$$
$$\leq\ distance\ to\ goal\ from\ x, y$$

Lecture 1: Search
Beam Search

- Beam = 2

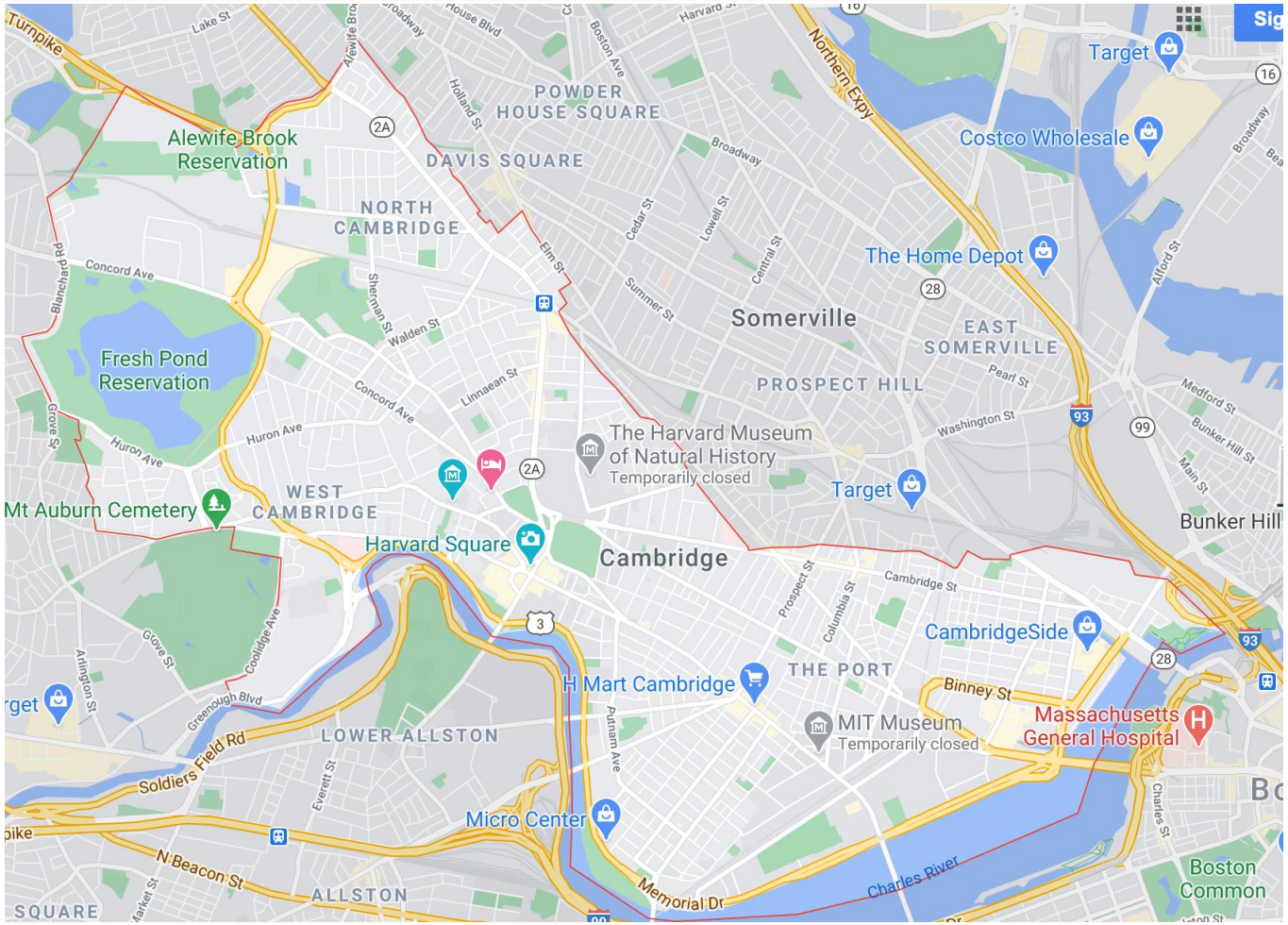


Lecture 1: Search
Beam Search

- Beam = 3
- *Machine translation, sequence to sequence*

Lecture 1: Search
A* VS Beam Search

Demo



Lecture 1: Adversarial Search

- **Adversarial search** the algorithm faces an opponent that tries to achieve the opposite goal.
- Often, AI that uses adversarial search is encountered in games, such as tic tac toe.

Lecture 1: Adversarial Search Minimax / Game

S_0 : initial state

PLAYER(s) : returns which player to move in state s

ACTIONS(s) : returns legal moves in state s

RESULT(s, a) : returns state after action a taken in state s

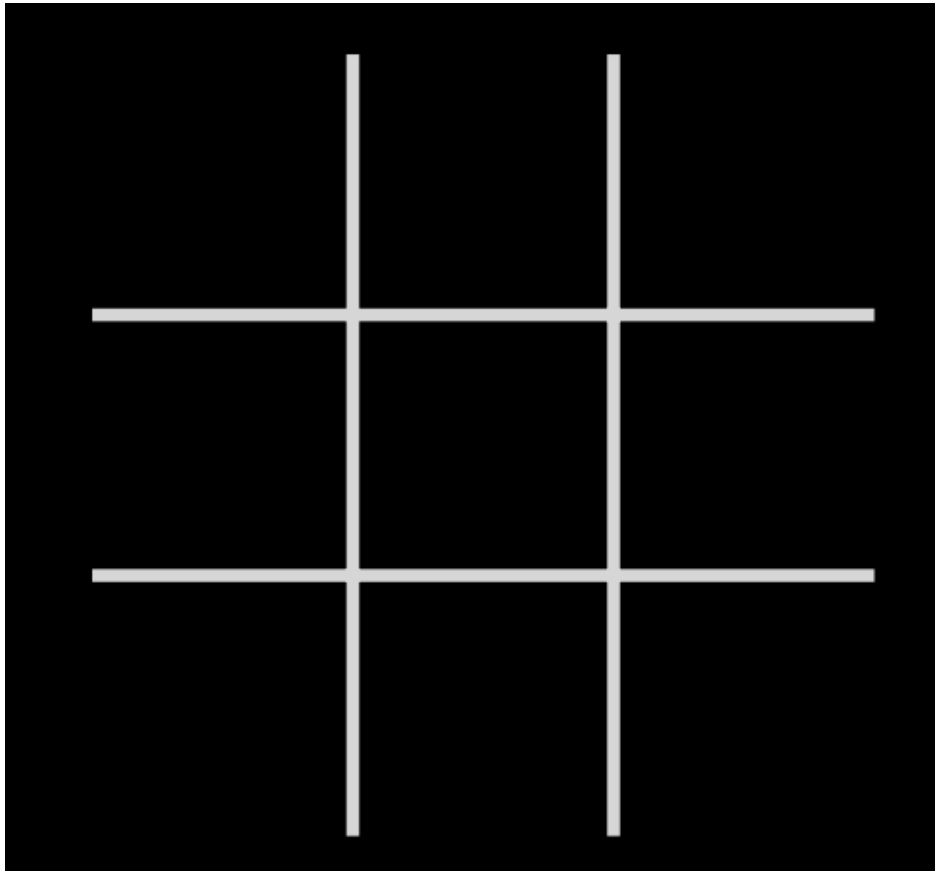
TERMINAL(s) : checks if state s is a terminal state

UTILITY(s) : final numerical value for terminal state s

Lecture 1: Adversarial Search Minimax / Game

Problem: don't know opponent's policy

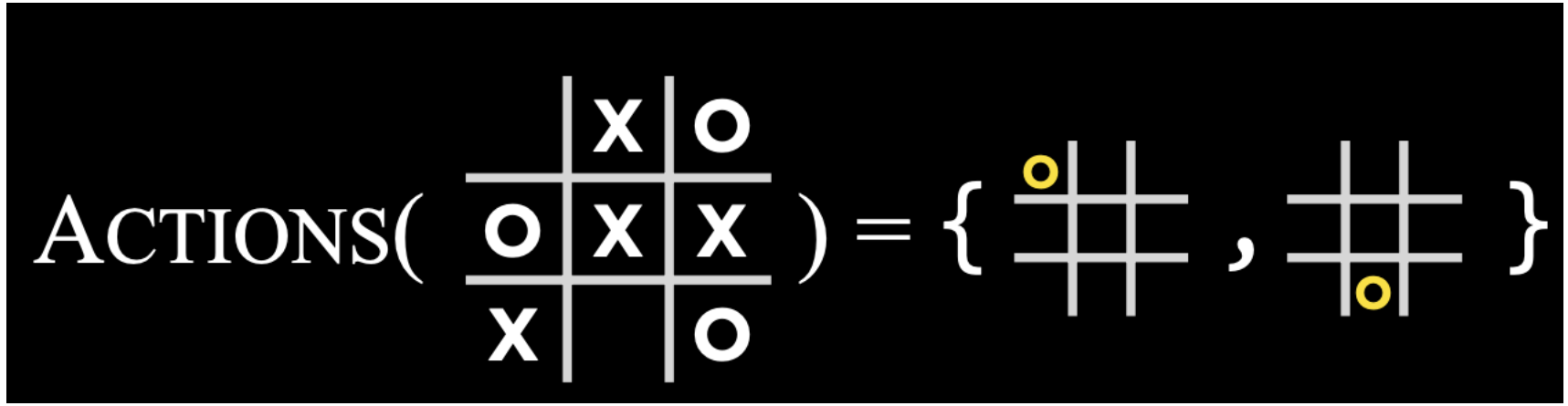
Approach: assume the worst case



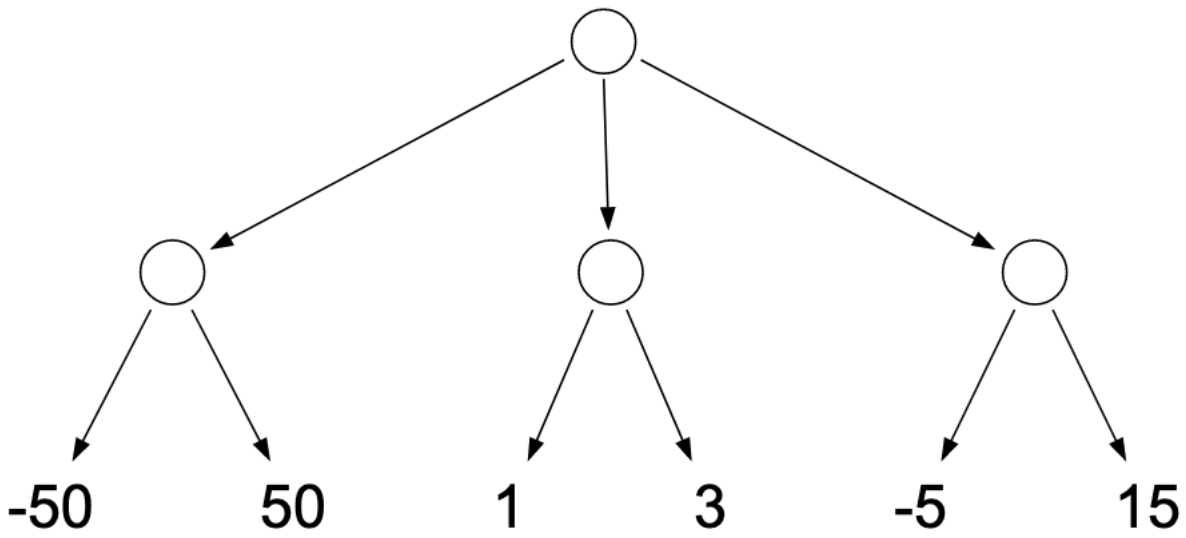
Lecture 1: Adversarial Search Minimax / Game

Problem: don't know opponent's policy

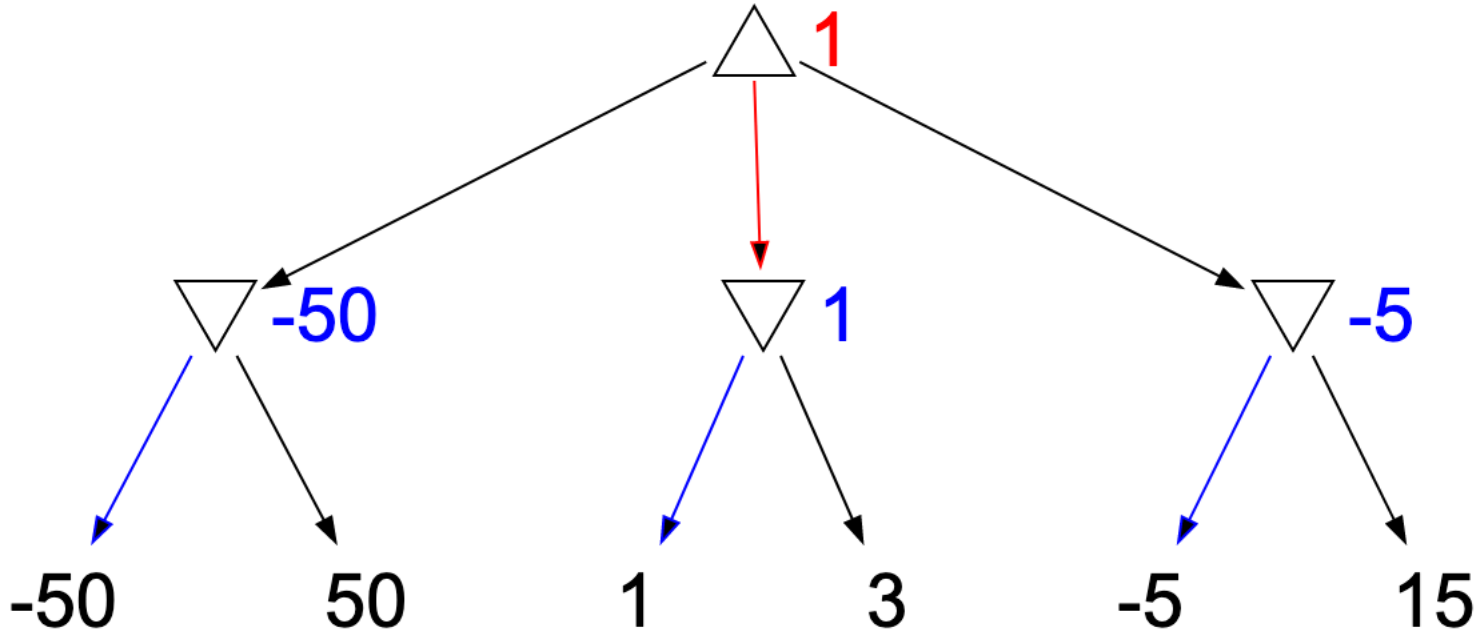
Approach: assume the worst case



Lecture 1: Adversarial Search Minimax / Game

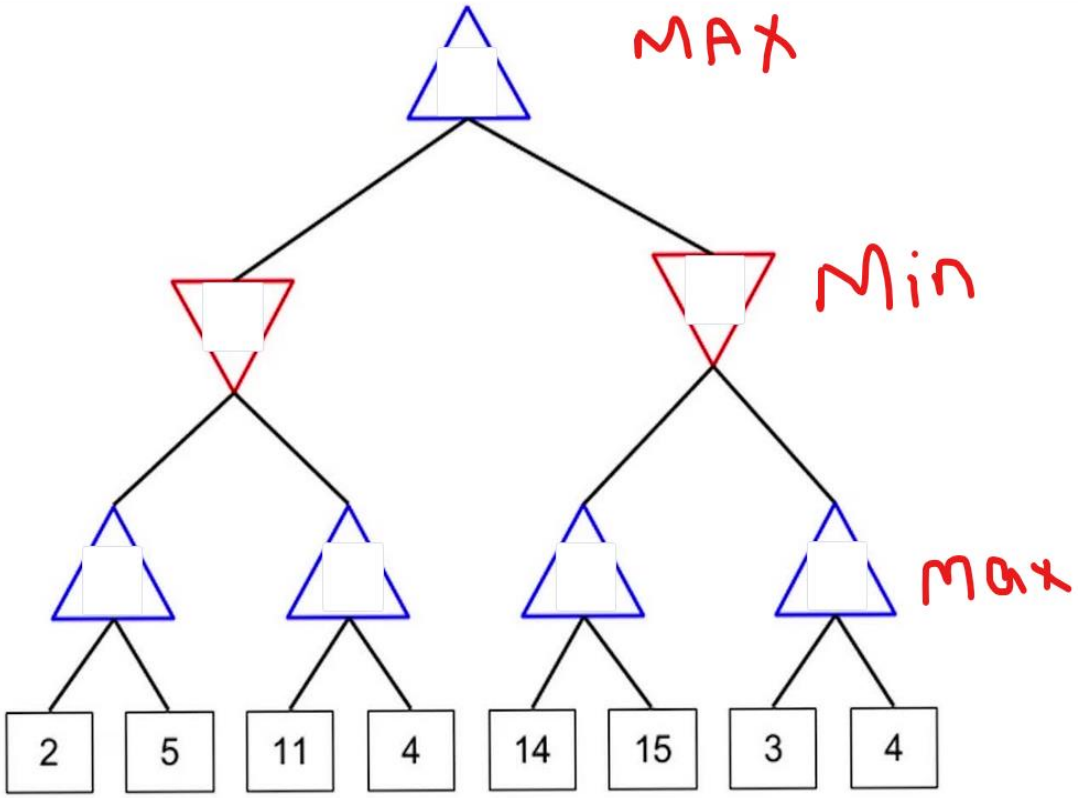


Lecture 1: Adversarial Search Minimax / Game



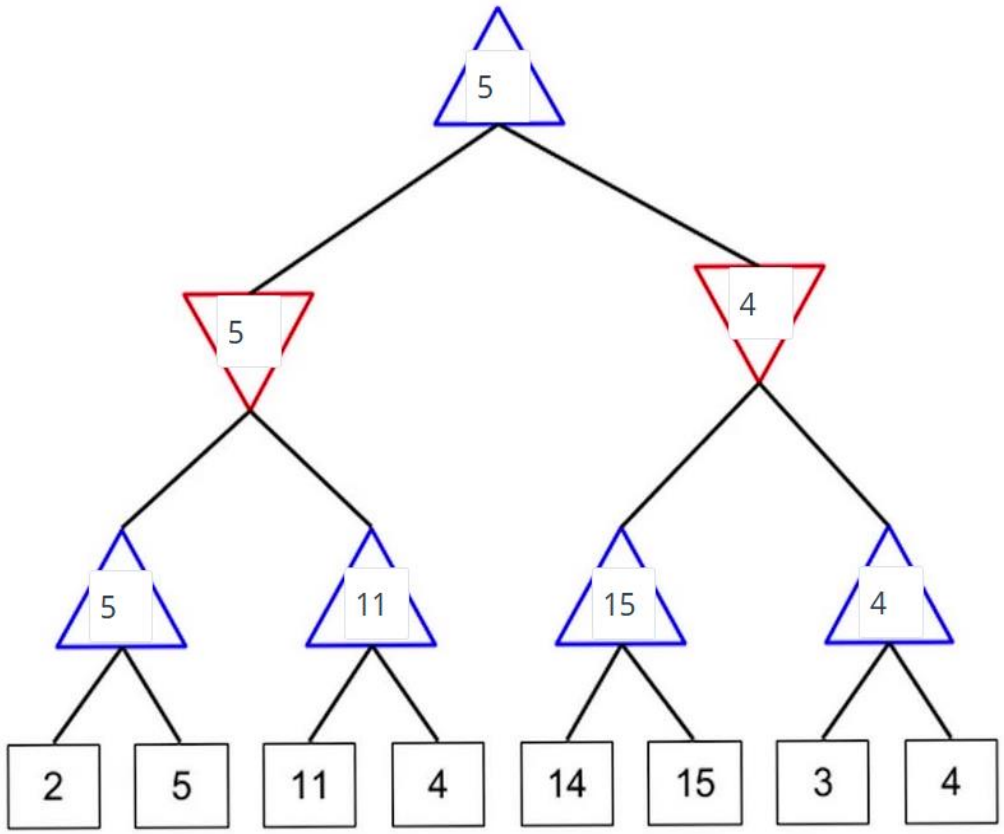
Lecture 1: Adversarial Search Minimax / Game – quiz

- Remember to fill in values bottom-up: Level 2 first, then Level 1 and finally the root node.
- Also note that MAX nodes are shown as upward triangles (△), and MIN nodes as downward triangles (▽).

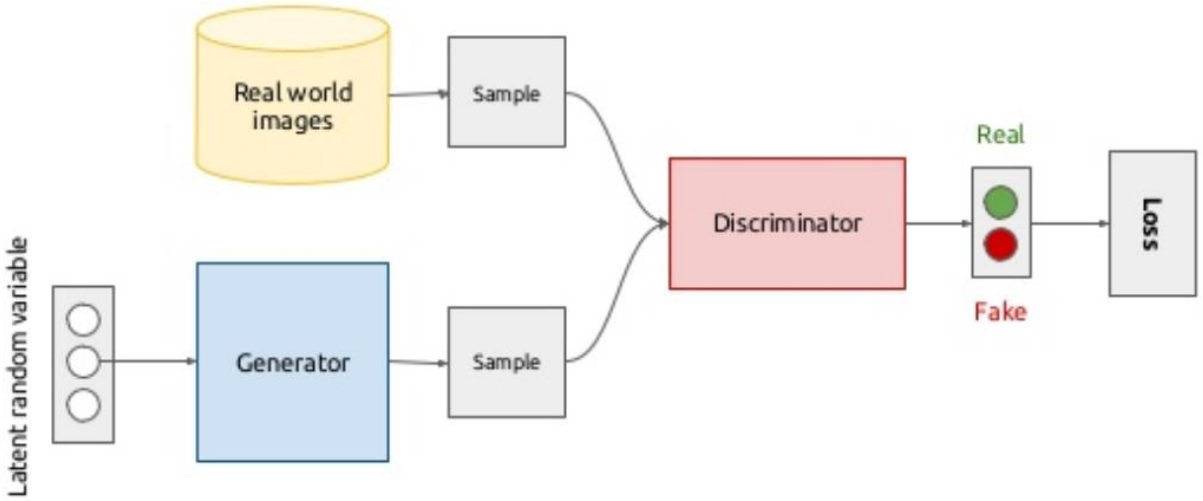


Lecture 1: Adversarial Search Minimax / Game – quiz

- Remember to fill in values bottom-up: Level 2 first, then Level 1 and finally the root node.
- Also note that MAX nodes are shown as upward triangles (△), and MIN nodes as downward triangles (▽).



Lecture 1: Adversarial Search
Minimax / GAN



$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

log prob of D predicting that real-world data is genuine

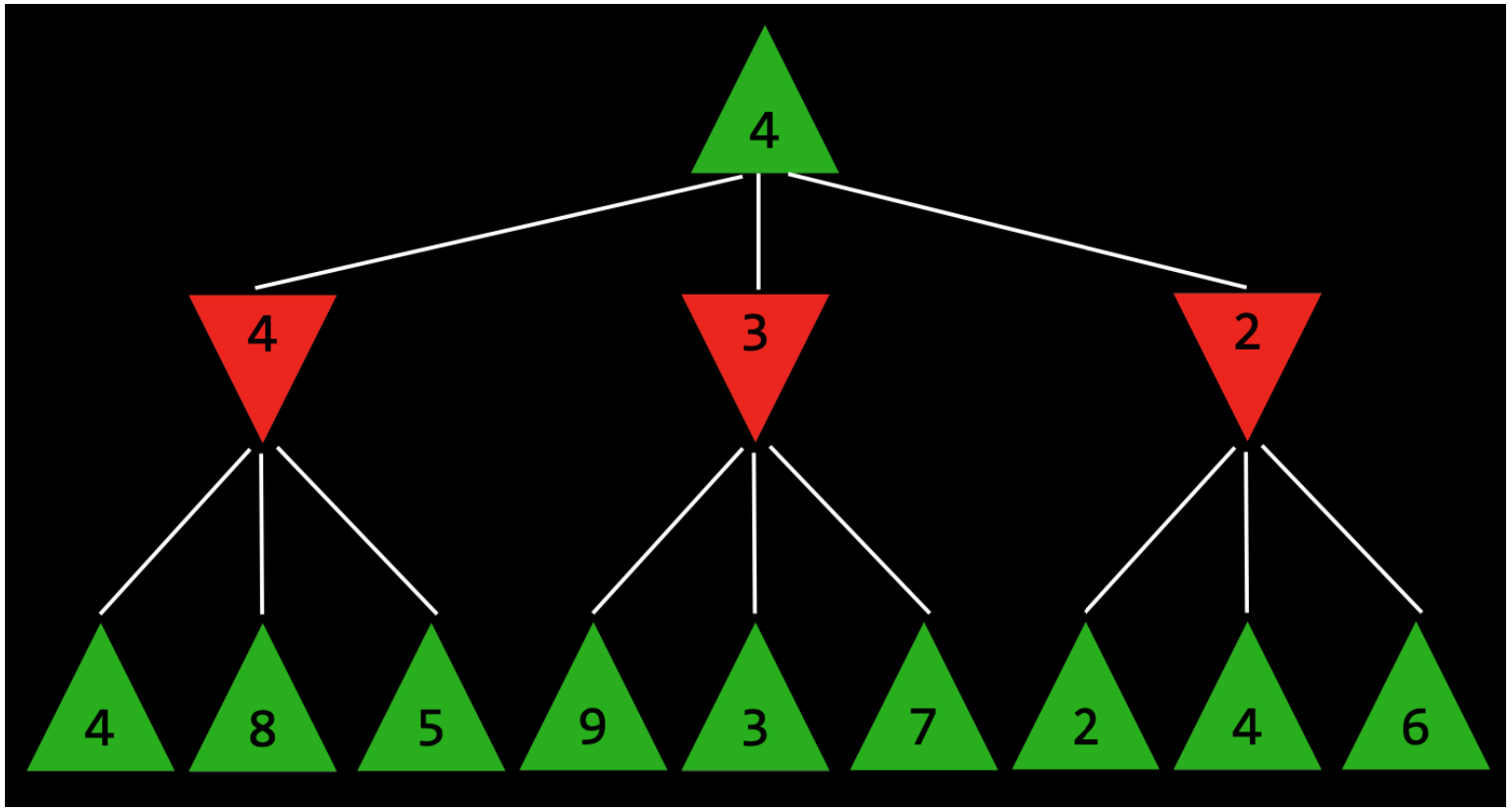
log prob of D predicting that G's generated data is not genuine

https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

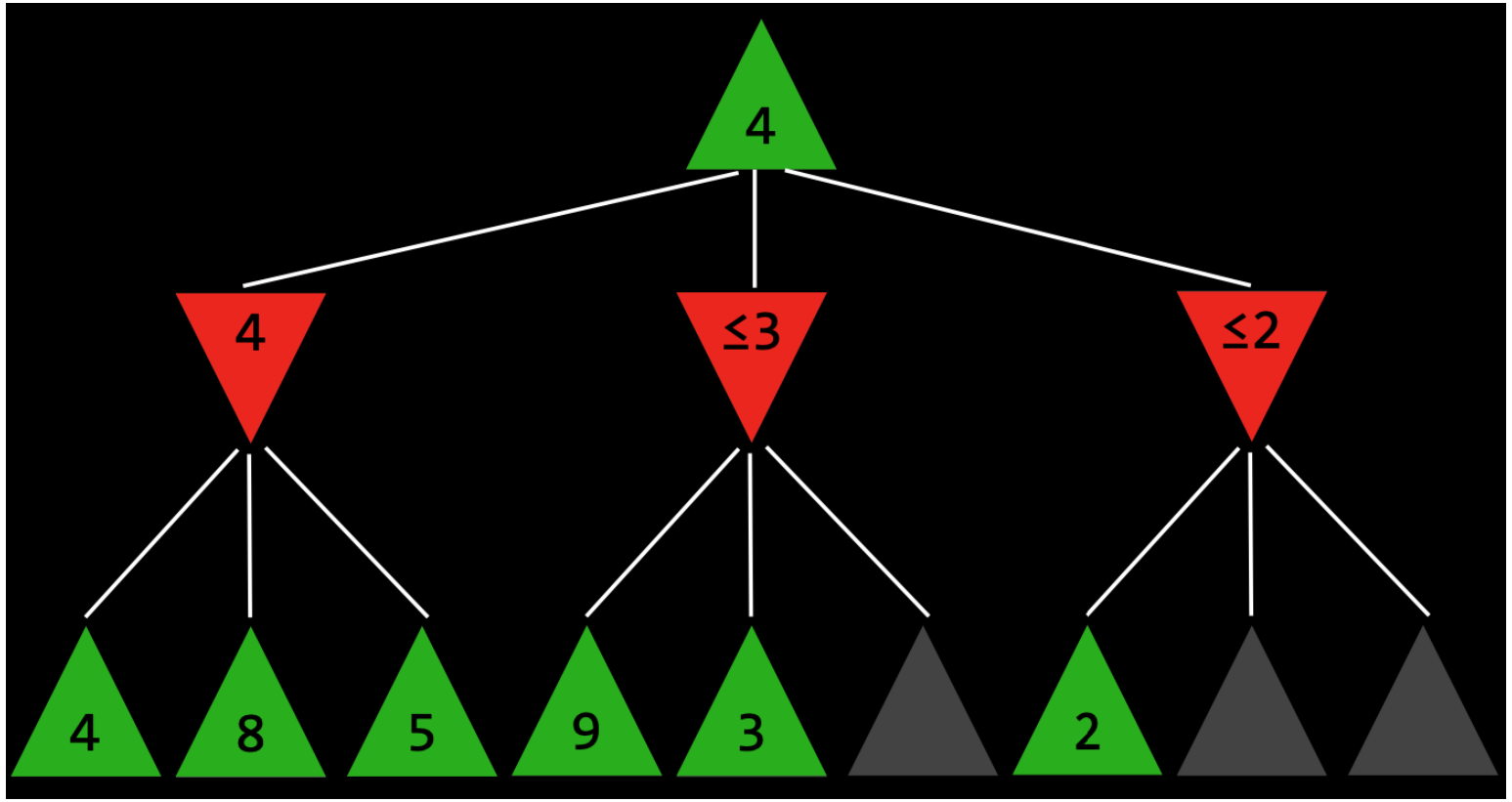
Lecture 1: Adversarial Search Minimax / Game

https://www.thispersondoesnotexist.com

Lecture 1: Adversarial Search
Alpha-Beta Pruning / Game

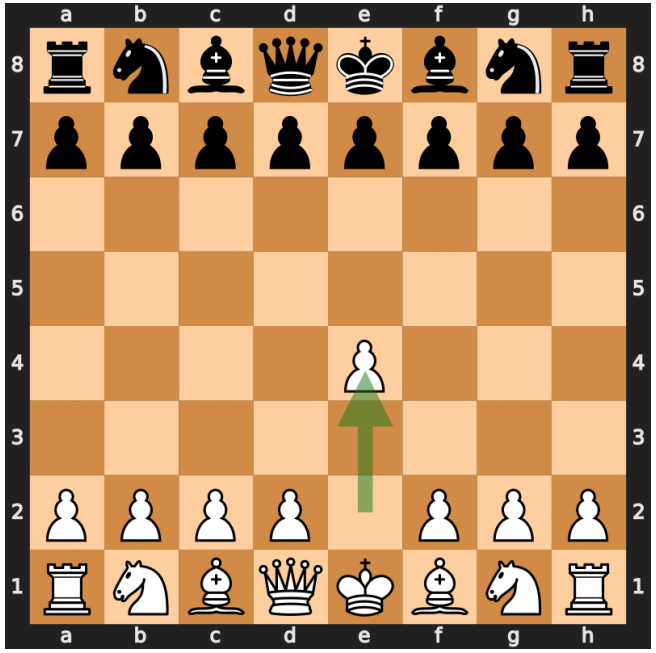


Lecture 1: Adversarial Search
Alpha-Beta Pruning / Game



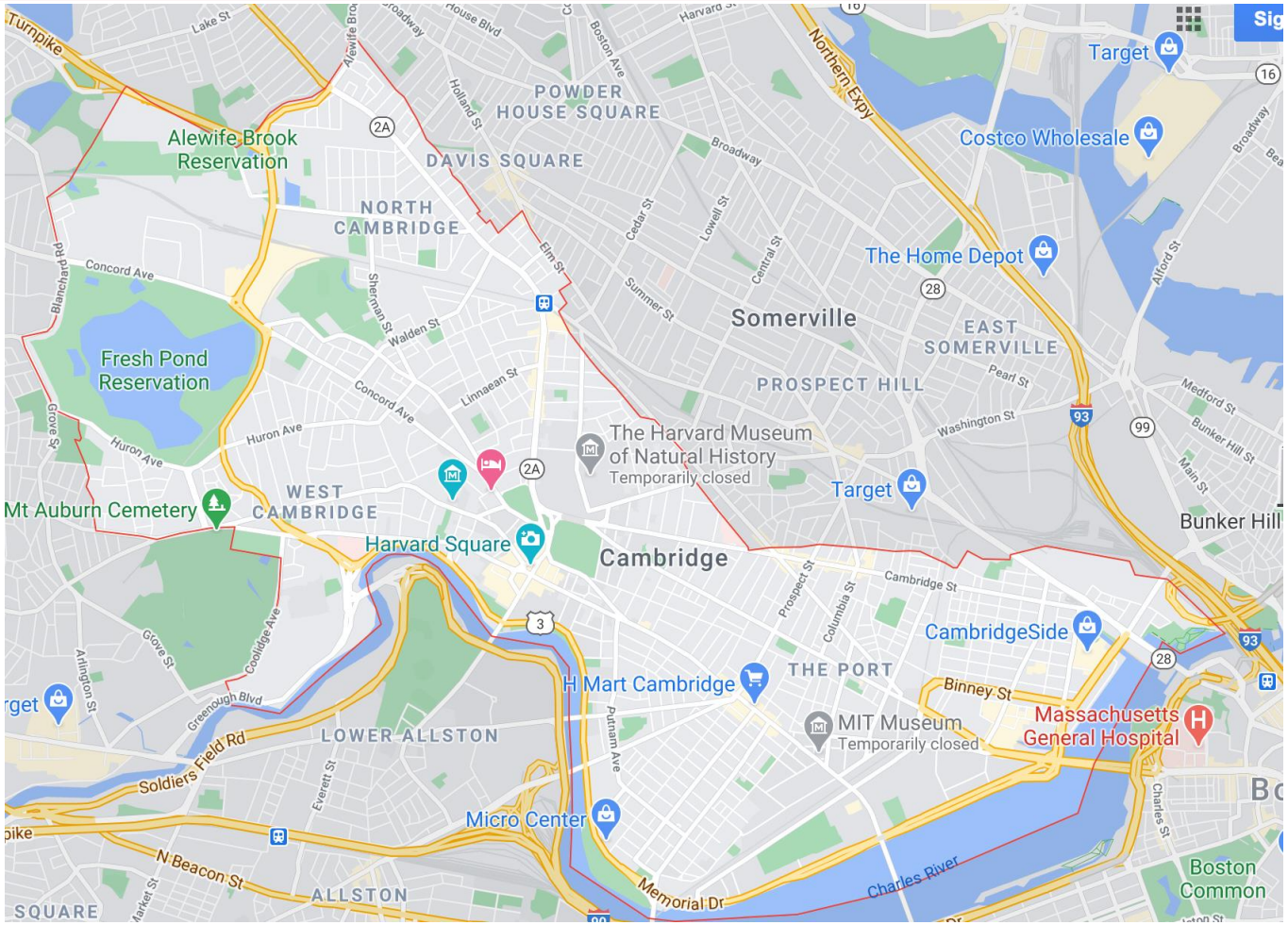
Lecture 1: Adversarial Search Alpha-Beta Pruning / Game

Deep Blue versus Kasparov



Lecture 1: Adversarial Search

Demo



Lecture 1: Adversarial Search Summary

- AI
- Search

Lecture 1: Adversarial Search
Next:

- Logic