

# Artificial Intelligence

**CSC 665**

*tyler dae devlin*

# Administrivia

- **Homework 0** ~~due yesterday~~ tonight at midnight.
- Reminder that the late policy allows you to submit up to **5 days** late with a 10% penalty per day.
- Homework 0 is free points!
- **Homework 1** is out on the course website

# **Search II**

***2.8.2024***

- **Search:** make decisions by looking ahead
- **Logic:** deduce new facts from existing facts
- **Constraints:** find a way to satisfy a given specification
- **Probability:** reason quantitatively about uncertainty
- **Learning:** make future predictions from past observations

# Recap

# Modeling (last time)

**Start state:**  $s_0 \in S$

**Possible actions:**  $\text{Actions}(s) \subseteq A$

**Action cost:**  $\text{Cost}(s, a) \in \mathbb{R}_{\geq 0}$

**Transition model:**  $\text{Succ}(s, a) \in S$

**Goal test:**  $\text{IsEnd}(s) \in \{\text{True}, \text{False}\}$

state space  $S$ , action set  $A$ , non-negative real numbers  $\mathbb{R}_{\geq 0}$

# Backtracking search (last time)

**Global state:** minimum cost path, set of explored (node, cost) pairs

**function** search( $s$ , path) :

- **if** IsEnd( $s$ ) :
  - update the minimum cost path
- **for each** action  $a \in \text{Actions}(s)$  :
  - **if** Succ( $s, a$ ) hasn't been explored at Cost( $s, a$ ) yet:
    - add (Succ( $s, a$ ), Cost( $s, a$ )) to the explored set
    - extend path with Succ( $s, a$ ) and Cost( $s, a$ )
    - recurse: search(Succ( $s, a$ ), path)

*Separating modeling from inference means we only have to write the inference code once.*



**More inference algorithms**

# Breadth-first and depth-first search

- Last time: **backtracking** search implemented **recursively**
- Today: **BFS** and **DFS** implemented **iteratively**
- Every iterative program can be implemented recursively, and vice-versa

# General approach

- Start with a frontier that contains  $s_0$ , and an empty explored set
- **While** the frontier is nonempty:
  - Pop a node  $s$  from the frontier
  - **If** IsEnd( $s$ ) : **return** solution
  - Add  $s$  to the explored set
  - Expand  $s$ , adding Succ( $s, a$ ) to the frontier **for** each  $a \in \text{Actions}(s)$ , as long as it's neither in the frontier nor already explored

# BFS vs. DFS

- **Breadth-first search (BFS)**
  - Expands the **shallowest** node in the frontier
  - Explores nodes in order of increasing depth
  - Frontier is a **queue** (FIFO)
- **Depth-first search (DFS)**
  - Expands the **deepest** node in the frontier
  - Equivalent to a backtracking search that stops after the first solution
  - Frontier is a **stack** (LIFO)

***[maze examples]***

# Two ways to analyze algorithms

- **Correctness**

- Exact or approximate?
- If approximately correct, how far off from exactness?
- If exactly correct, under what conditions?

- **Efficiency**

- Asymptotic analysis (big-oh)
- Time
- Space

# Correctness of search algorithms

- **Backtracking search:** returns shortest path for **any** cost function
- **BFS:** returns shortest path for (non-negative) **constant** cost function
- **DFS:** returns shortest path for **zero** cost function

# Efficiency of search algorithms

- **Backtracking search:**  $O(D)$  space,  $O(b^D)$  time
- **BFS:**  $O(b^d)$  space,  $O(b^d)$  time
- **DFS:**  $O(D)$  space,  $O(b^D)$  time

$b$  actions per state, solution depth  $d$ , maximum depth  $D$



# Summary

<i>algorithm</i>	<i>cost function</i>	<i>space</i>	<i>time</i>
backtracking	any	linear	exponential
BFS	constant	exponential	exponential
DFS	zero	linear	exponential

# Layered search

- BFS works because it explores in **layers** of equal depth
- But only if the cost function is **constant**
- Can we make the idea of a layered search work with **non-constant** action costs?

Yes, thanks to Dijkstra!

# Uniform Cost Search (UCS, Dijkstra's Algorithm)

- Start with a frontier that contains  $s_0$ , and an empty explored set
- **While** the frontier is nonempty:
  - Pop the node  $s$  with smallest priority  $p$  from the frontier
  - **If** IsEnd( $s$ ) : **return** solution
  - Add  $s$  to the explored set
  - **For** each  $a \in \text{Actions}(s)$ ,
    - Get  $s' = \text{Succ}(s, a)$
    - **If**  $s'$  is already explored: **continue**
    - Add  $s'$  to frontier with priority  $p + \text{Cost}(s, a)$