CSE 240
Section 3

January 19, 2022

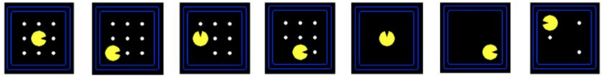
Topics To Remember!

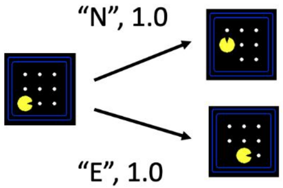
- Search Problems (L2, p 11)
- What’s in a State Space (L2, p15)
- DFS (L2, p27-30)
- Backtracking (L2, p32-33)
- BFS (L2, p34-38)
- Comparison between search algos (L3, p6)
- UCS (L3, p9 -15)
- Heuristics/A* (L3, p17-26)
- Consistent Heuristics (L3, p27)
- Relaxation (L4, p23-25)
- Minimax example (L5, p21)
- Alpha-Beta Pruning

L2

Search Problems

- A search problem consists of:
  - A state space
  - A successor function (with actions, costs)
  - A start state and a goal test



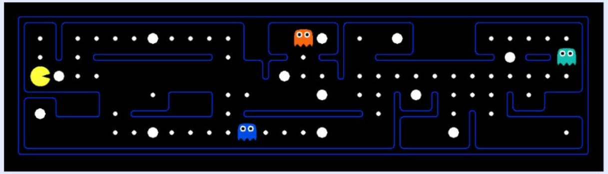


- A solution is a sequence of actions (a plan) which transforms the start state to a goal state

L2

What's in a State Space?

The world state includes every last detail of the environment



A search state keeps only the details needed for planning (abstraction)

- Problem: Finding a path
  - States: (x,y) location
  - Actions: N, S, E, W
  - Successor: update location only
  - Goal test: is (x,y)=END

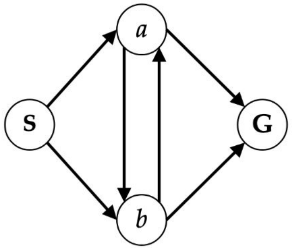
- Problem: Eat-All-Dots
  - States: {(x,y), dot Booleans}
  - Actions: N, S, E, W
  - Successor: update location and possibly a dot Boolean
  - Goal test: dots all false

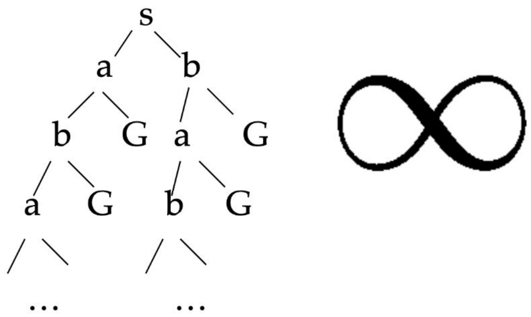
L2

State Space Graphs vs. Search Trees

Consider this 4-state graph:

How big is its search tree (from S)?





Important: Lots of repeated structure in the search tree!

L3

Comparison

Algorithms	Action Costs	Space	Time
Backtracking	Any	$O(bm)$	$O(b^m)$
DFS	Zero	$O(bm)$	$O(b^m)$
BFS	Constant > 0	$O(b^s)$	$O(b^s)$
ID-DFS	Constant > 0	$O(bs)$	$O(b^s)$

L4

General Framework

Definition: relaxed search problem

A **relaxation** P_{rel} of a search problem P has costs that satisfy:

$$\text{Cost}_{\text{rel}}(s, a) \leq \text{Cost}(s, a).$$

Definition: relaxed heuristic

Given a relaxed search problem P_{rel}, define the **relaxed heuristic** $h(s) = \text{FutureCost}_{\text{rel}}(s)$, the minimum cost from s to an end state using $\text{Cost}_{\text{rel}}(s, a)$.

L4

Tradeoff

- Efficiency
  - $h(s) = \text{FutureCost}_{rel}(s)$ must be easy to compute
  - Closed form, easier search, independent subproblems
- Tightness
  - heuristic h(s) should be close to FutureCost(s)
  - Don’t remove too many constraints

Alpha-Beta Pruning

- Alpha cut (prune max node)
- Beta cut (prune min node)