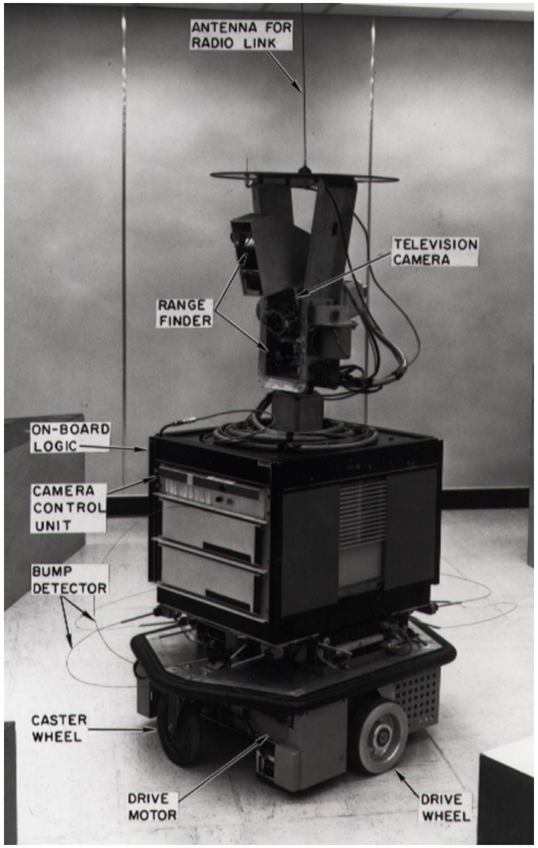
Lecture 5:

Minimax Search

4 Feburary 2020

A* Search

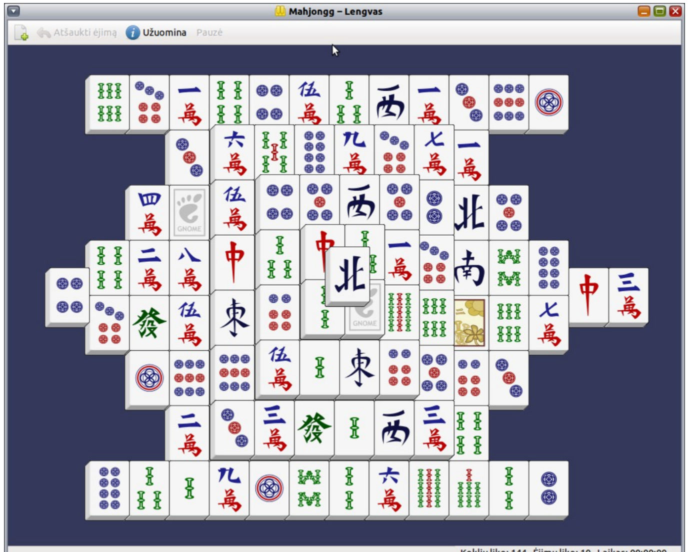
- Order nodes by sum of path length and heuristic
- Priority queue order by path(s) + h(s)
- Will find optimal path if using admissible heuristic



priority_queue.push(initial state, 0)
While(!priority_queue.empty())
    s = priority_queue.pop()
    if(s == goal)
        return
    if (!s.visited)
    - s.visited=true
    - for all actions $a \in A$
        $s' = \tau(s,a)$
        $s'_{path\_length} = s_{path\_length} + length(a)$
        priority_queue.push(s', $s'_{path\_length} + h(s')$)

Admissible Heuristics

- A heuristic is admissible if it never overestimates the cost from a node to the goal
  - $h(s) \leq cost(path(s,s_{goal}))$
  - Multiple goals or paths to goal
    - h(s) less than minimum cost over all paths to goals
    - Example: Find a path to the nearest post office
      - Doesn't matter what post office you get to
      - h(s) must be less then cost of shortest path to nearest post office
- Examples
  - Robot in plane
    - Euclidean distance
  - Robot in grid
    - Manhattan distance
  - Mahjong Solitaire
    - 1/2 * number of tiles left



Adversarial Search

Can we apply A* to chess?

What makes adversarial search difficult?

Adversarial Search

- Adversarial problems
  - Agent is competing against another agent
    - Could be a human or another AI agent
  - Agents are trying to accomplish conflicting tasks
    - Trying to stop other agent from reaching its goal
    - Other agent trying to stop you from reaching your goal
- Games
  - Chess
  - Checkers
  - tic tac toe
  - Computer games
    - Dota 2
- What makes adversarial problems difficult?
  - Need to predict actions of other agent

O		X

What move will opponent make?

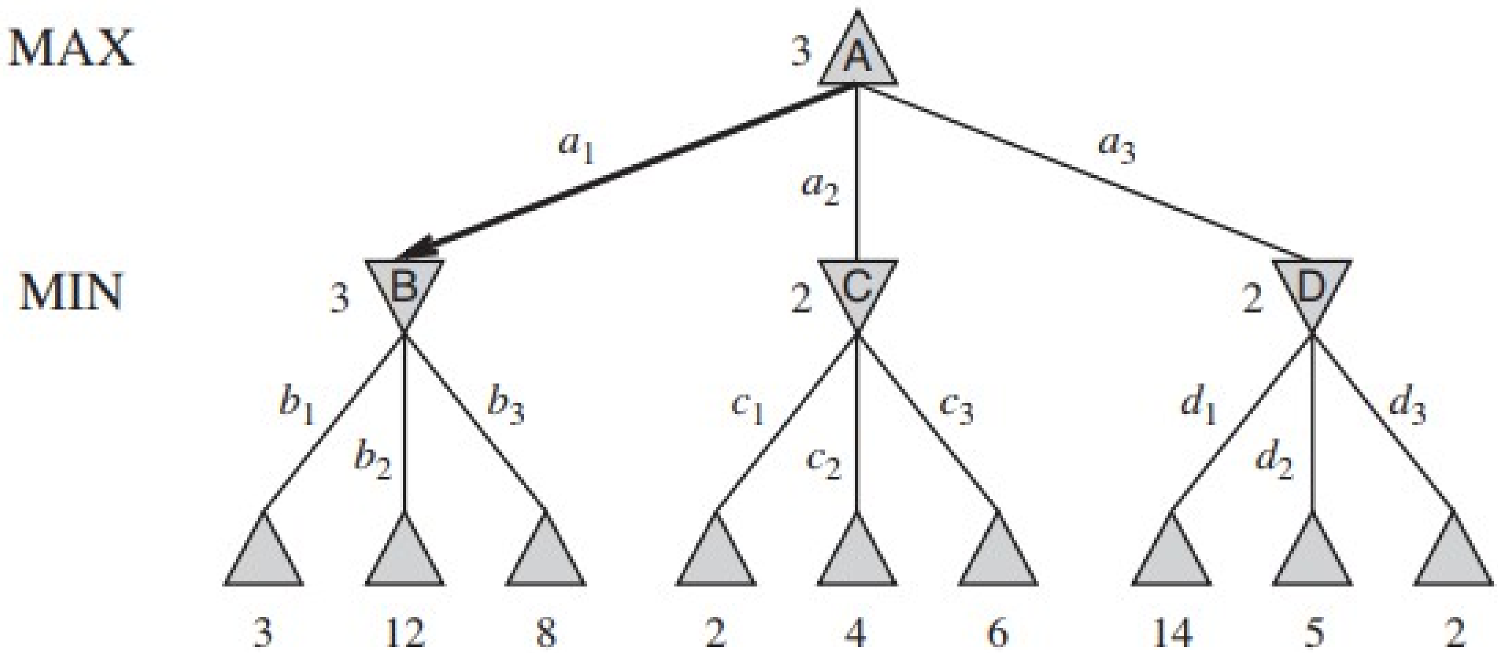
Adversarial Search

- Observation
  - A good state for you is a bad state for your opponent
  - A bad state for you is a good state for your opponent
  - Assume opponent will always pick worst state for you
  - Strategy:
    - Select action where opponent's best move will lead to best state for you
    - Select action with best worst state
    - Example: Chess
      - Opponent can mate in one
      - Select move that stops check-mate

Minimax search

```
function minimax(s, d, bMaxPlayer)
    if d = 0 or goal(s)
        return h(s)
    if bMaxPlayer then
        v := −∞
        for each action a
            s' = τ(s,a)
            v := max(v, minimax(s, d − 1, FALSE))
        return v
    else
        for each action a
            s' = τ(s,a)
            v := min(v, minimax(s, d − 1, TRUE))
        return v
```

Minimax search example



Example: tic tac toe

- Agent is O
- Agent moves next

O		X
	X	
		O

Minimax search: discussion

- How could you adapt minimax search to games with multiple adversaries
  - Poker
  - Chinese checkers

- How could you adapt minimax search to cooperative games?
  - Euchre or Bridge
- Problem's where other agent trying to accomplish unrelated task?
  - Two robot's in an environment

Decision Trees

- Nodes are states
  - At each state you need to make a decision specific to that state
  - Edge for each choice
  - Edge leads to state obtained by selecting choice
- Effectively, actions are choices
  - Different from other examples because actions are different for each state
- Example - navigating a road map
  - At intersection of Main and Oak
    - Turn right on Oak
    - Turn Left on Oak
    - Stay on Main Street
  - Options available different for different intersections