Uninformed Search

CSE 4617: Artificial Intelligence

Isham Tashdeed
Lecturer, CSE

Machines that are Intelligent

- Some have defined intelligence through the lens of **rationality** while others see it compared with **human** performance

- Another way to categorizing intelligence would be to consider intelligence being a property of **internal thought process** while others classify intelligence on **behavior**

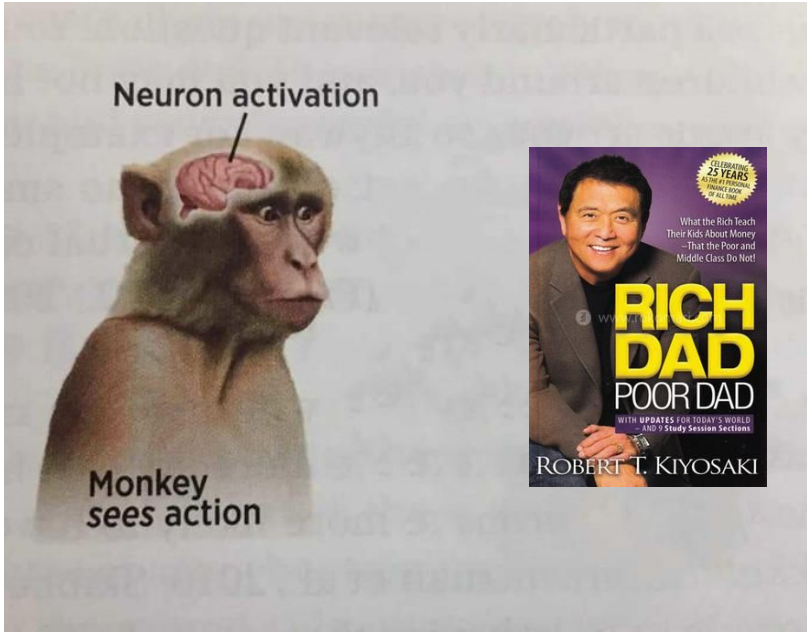
There are *four* combinations of these schools of thought.

Think Rationally

- Rationally → Doing the *right* thing
- What are the rules that govern correct thought?
- Let us infuse those rules into machines

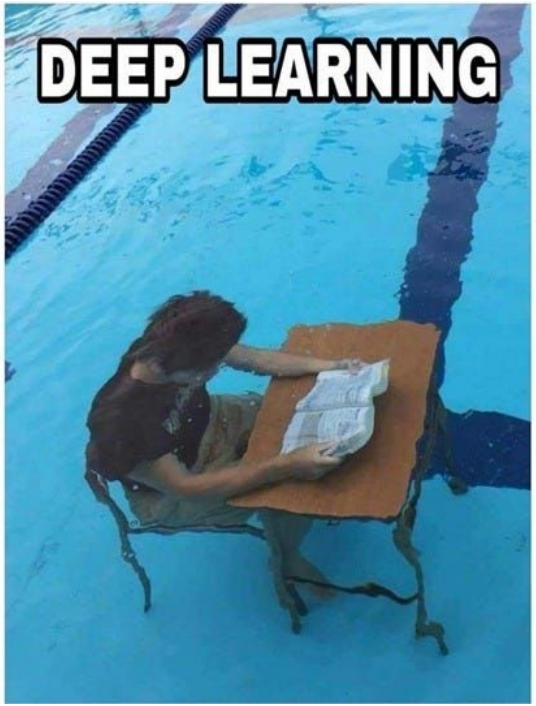
Think like people

- Reverse engineering the brain
- What are the processes going on inside our head
- Cognitive science



Act like people

- The Turing Test approach
- Almost all of modern Deep Learning advancements
- Fooling people by acting like them
- Tells us more about people than intelligence



Act rationally

- Focusing on the decisions made by the system
- Rational → Optimal
- System *acts* as to achieve the best possible outcome

"Big Data", "AI powered", "Prompting", "Agentic Approach"

Just behaves rationally

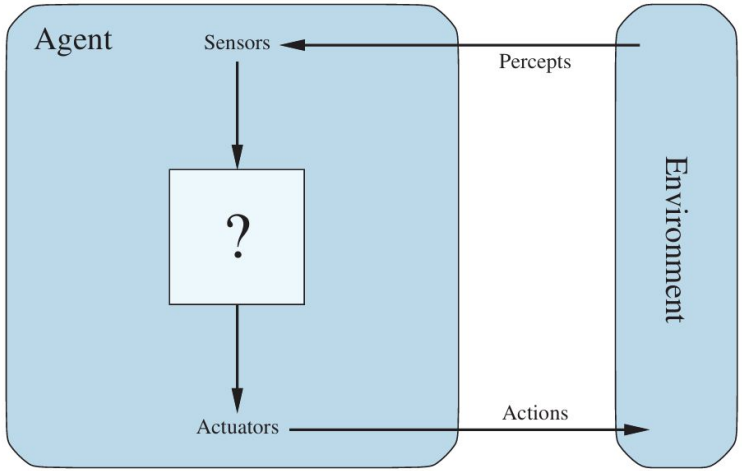
Rational Decisions

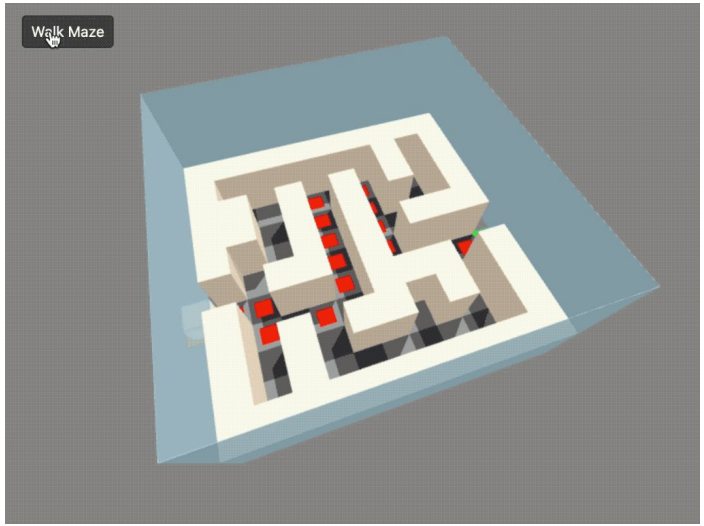
- Maximally achieve a pre-defined goal
- We only care about the actions → Not the thoughts / rationale behind those actions

Maximize your expected utility

Rational Agents

- Agent → An entity that *perceives* and *acts*
- Rational Agent → Selects the actions that would maximize the utility





Reflex Agents

- Choses actions based on only current perception / memory
- Does not consider the future consequences of its action
- May have a memory / model of the world
- Only considers how the world is currently, rather than what it could be after the action
- Pros → Very quick

But is it rational?

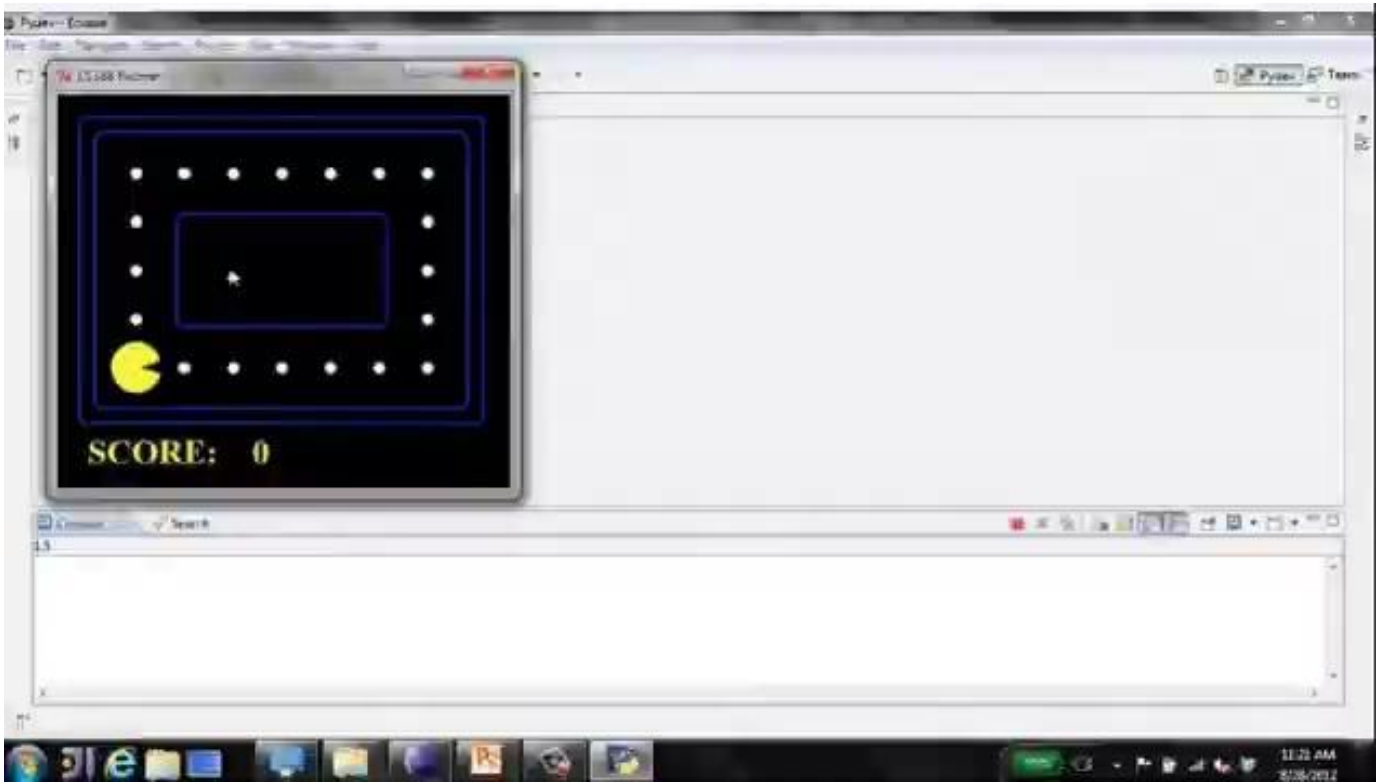
Reflex Agents

- Choses actions based on only current perception / memory
- Does not consider the future consequences of its action
- May have a memory / model of the world
- Only considers how the world is currently, rather than what it could be after the action
- Pros → Very quick

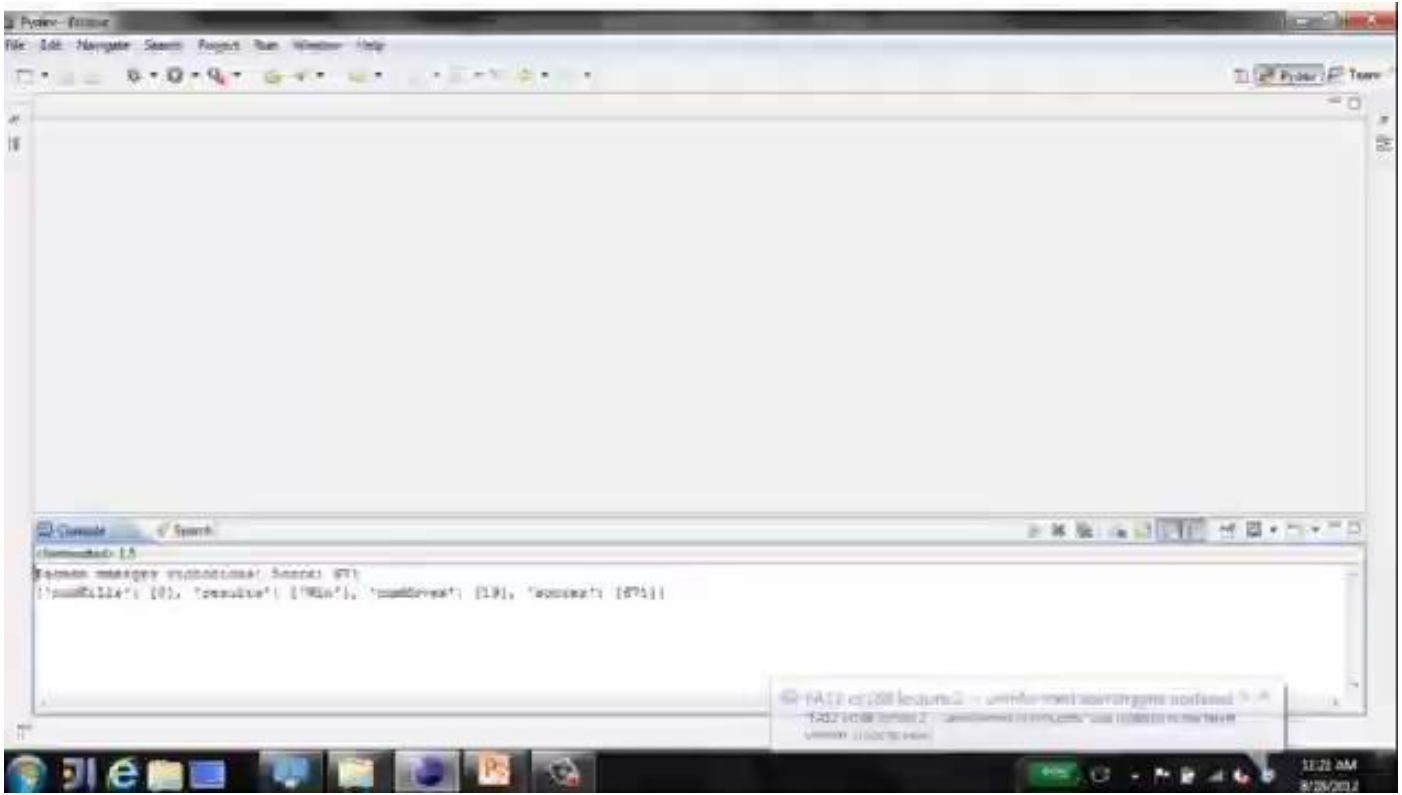
But is it rational?

- We only consider the outcome of the action, so it can be rational

Reflex Agents

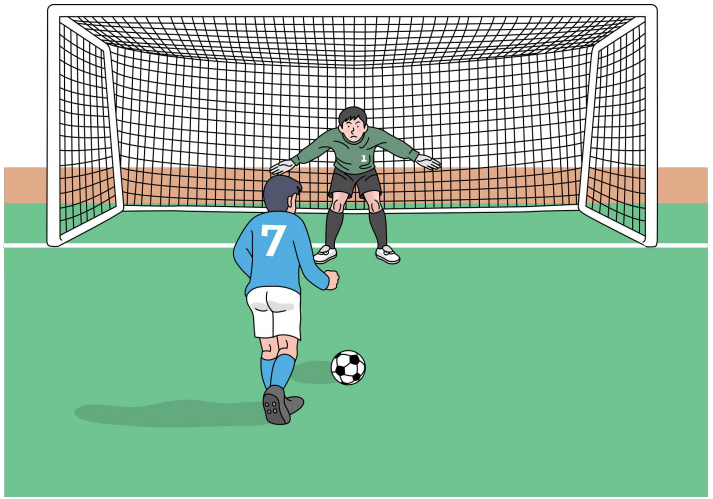


Reflex Agents

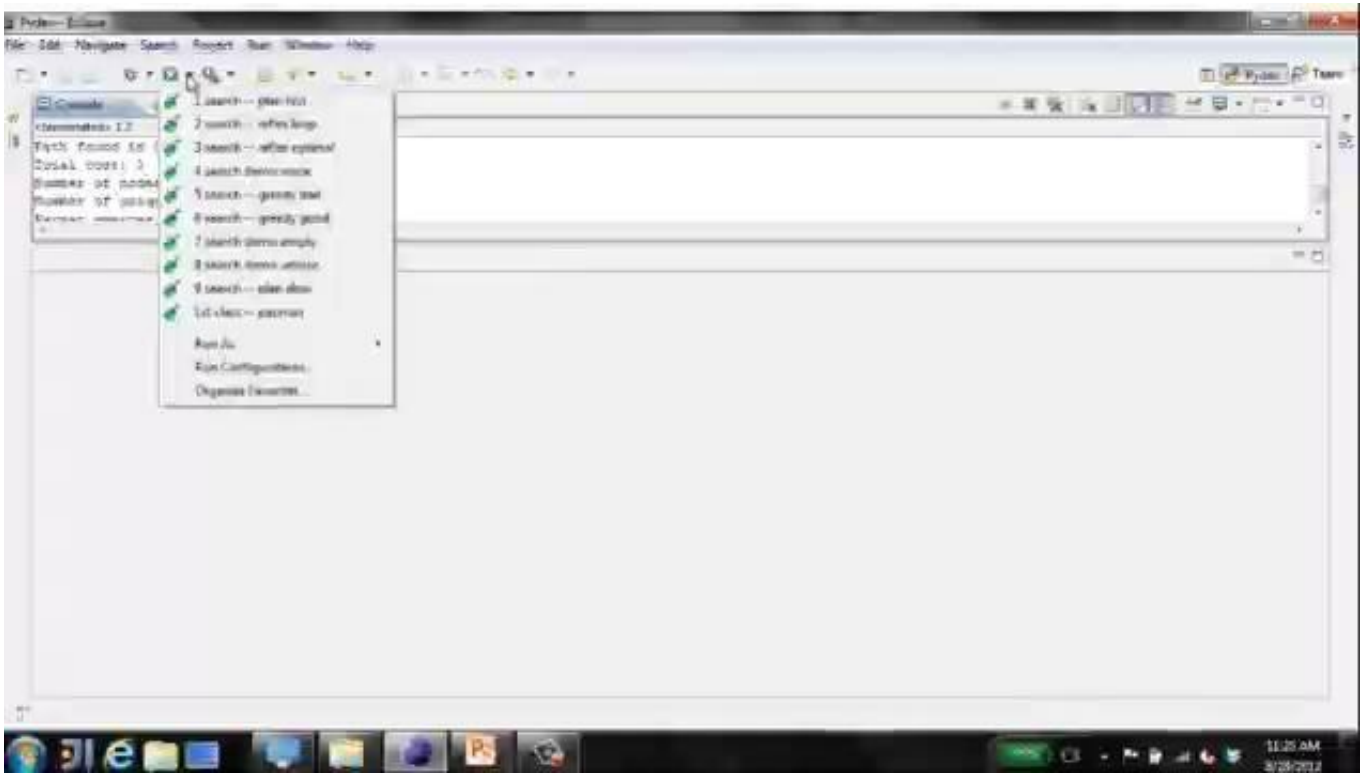


Planning Agents

- Takes action based on consequences (hypothesized) of the action
- **Must** have a model of the world. The model has to be updated as a response to the action taken
- Must formulate a goal
- A planning agent must be:
  - Optimal
  - Complete



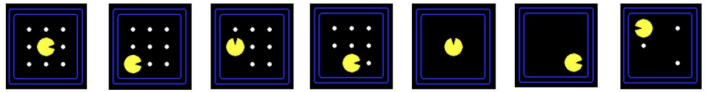
Planning Agents



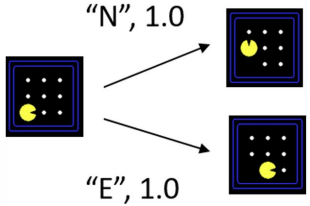
Search Problems

Defining a Search Problem

- **State space** → Set of all possible scenarios

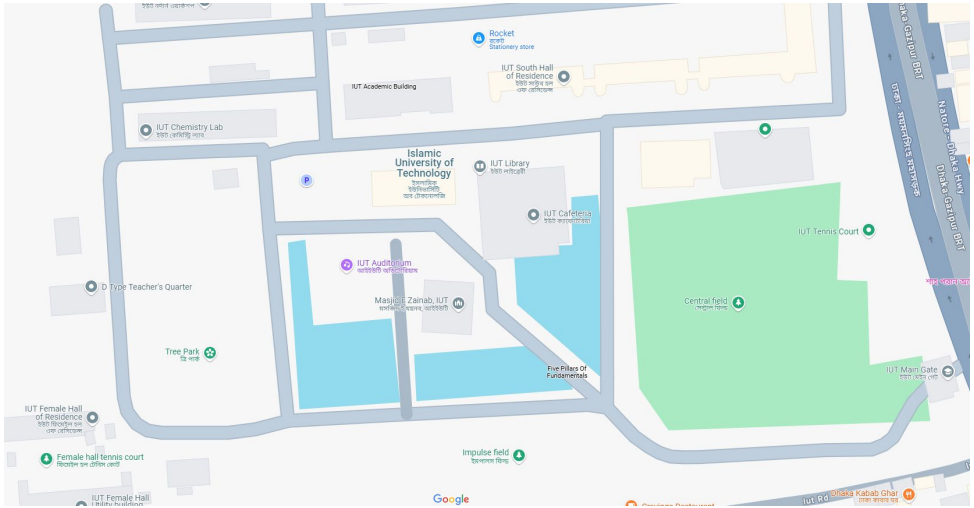


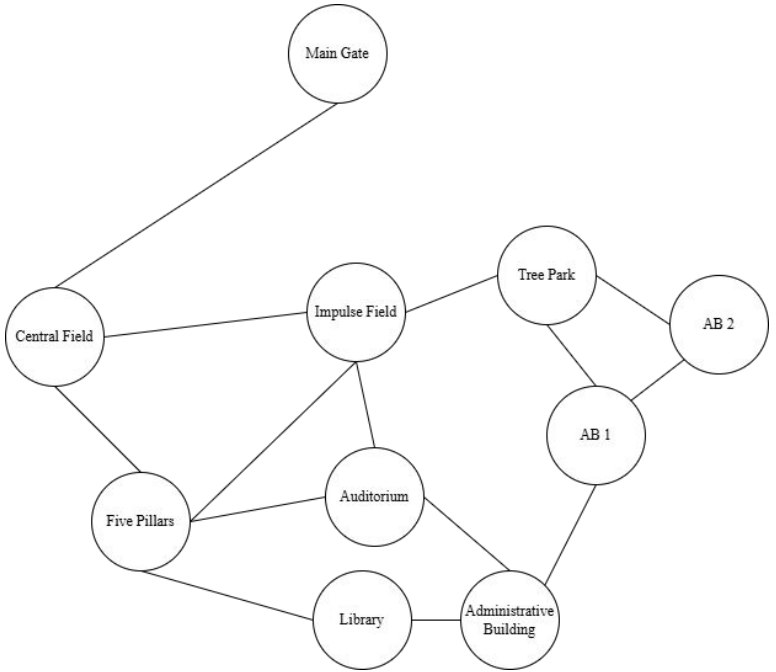
  - A singular state defines the agent's condition at the current time
  - Actions can take an agent from one state to another
- **Successor function** → Provides consequent states
  - Takes an action and provides possible states from current state
  - May take *cost* into consideration (if applicable)



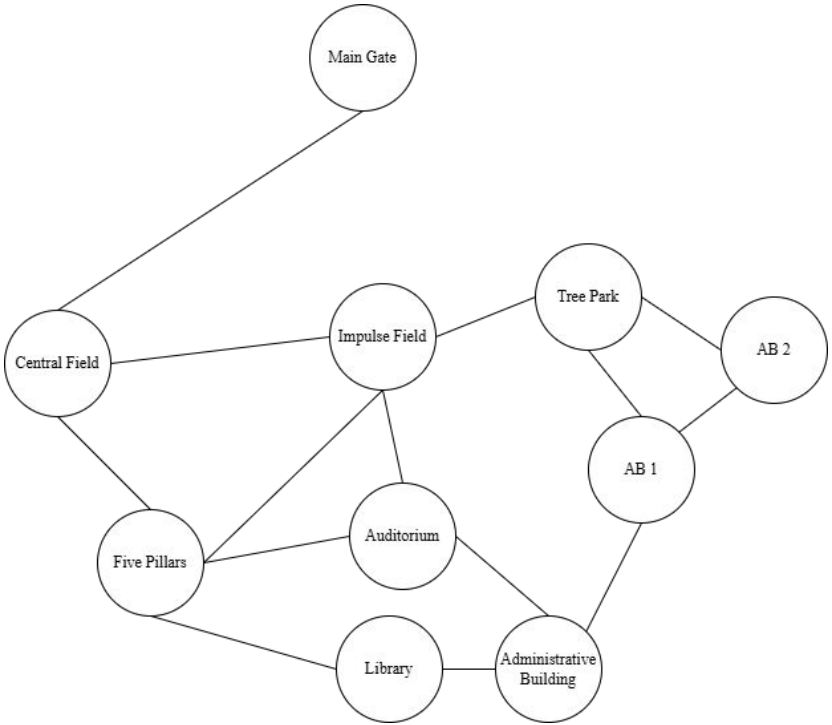
- **Start state**
- **Goal test**
- **Solution** → Sequence of actions, starting from start state to goal state

Example: Getting around IUT



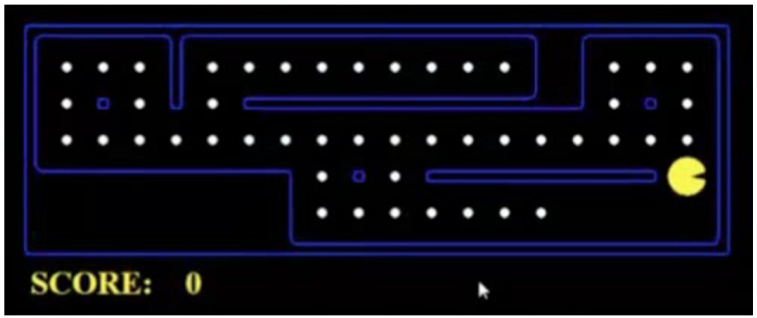


Example: Getting around IUT



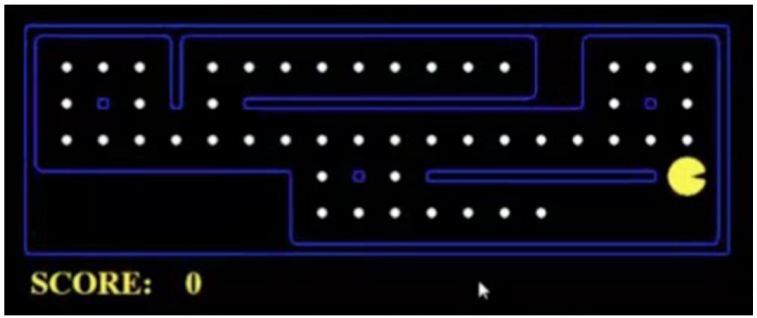
- State space?
  - Locations
- Successor Function?
  - Connections / Roads
- Start state?
  - Main gate
- Goal test?
  - Current state == AB2 ?
- Solution?
  - Path from start to goal state

Example: Maze Traversal



- State space?
- Successor Function?
- Start state?
- Goal test?
- Solution?

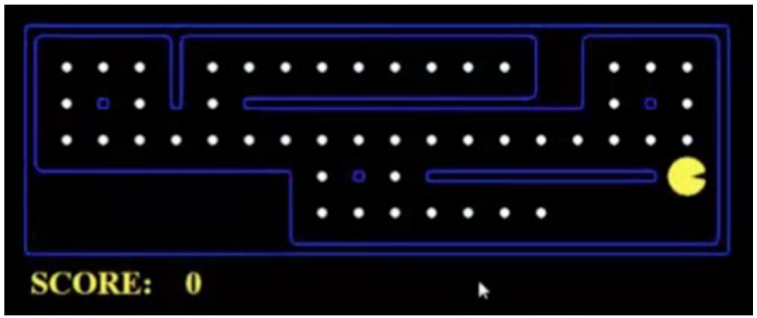
Example: Maze Traversal



- State space?
  - (x, y) location
- Actions: NSEW
- Successor Function?
  - Update the location of the agent
- Start state?
- Goal test?
  - Current (x, y) == END?

Only have to think about solving the maze

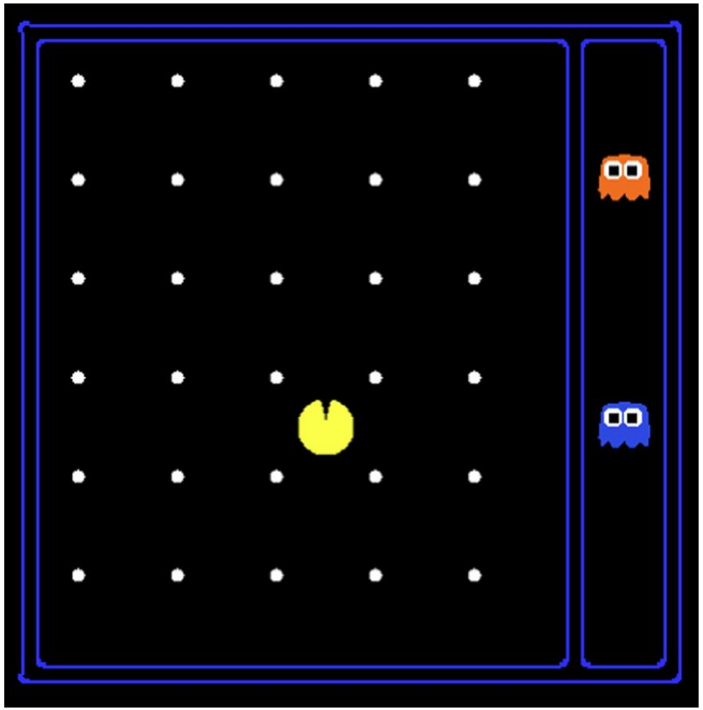
Example: Eat All Food



- State space?
  - (x, y) location, Food locations
- Actions: NSEW
- Successor Function?
  - Update the location and food
- Start state?
- Goal test?
  - All food eaten?

Have to think about solving the maze and eating all the food pellets

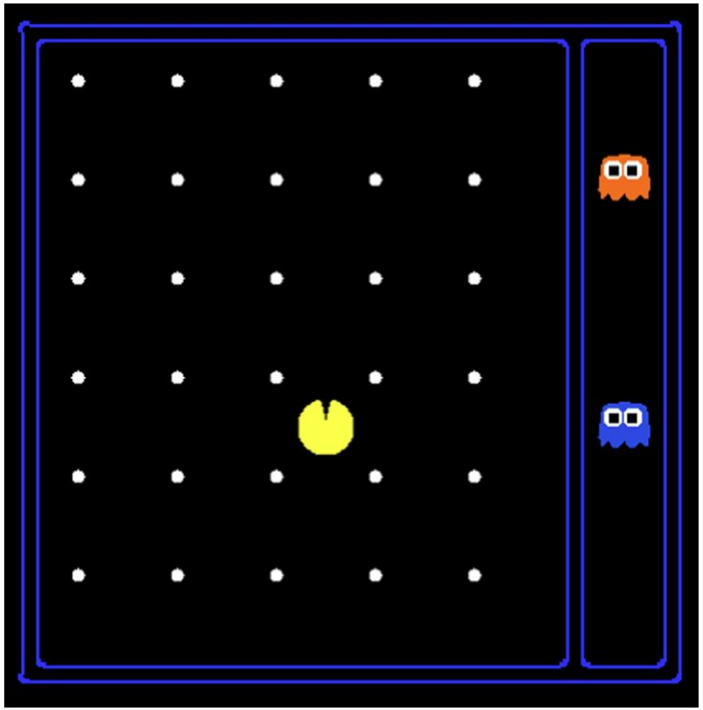
Size of the State Space



- Agent positions: 120
- # of food: 30
- # of ghosts: 2
- Ghost positions: 12
- Agent can face: NSEW

Total World states?

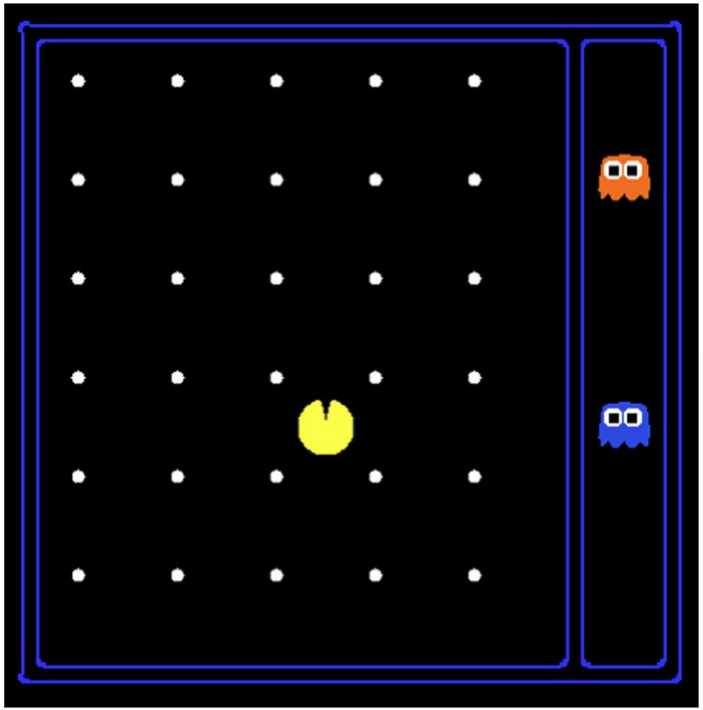
Size of the State Space



- Agent positions: 120
- # of food: 30
- # of ghosts: 2
- Ghost positions: 12
- Agent can face: NSEW

Total World states? → $120 * 2^{30} * 12^2 * 4$

Size of the State Space



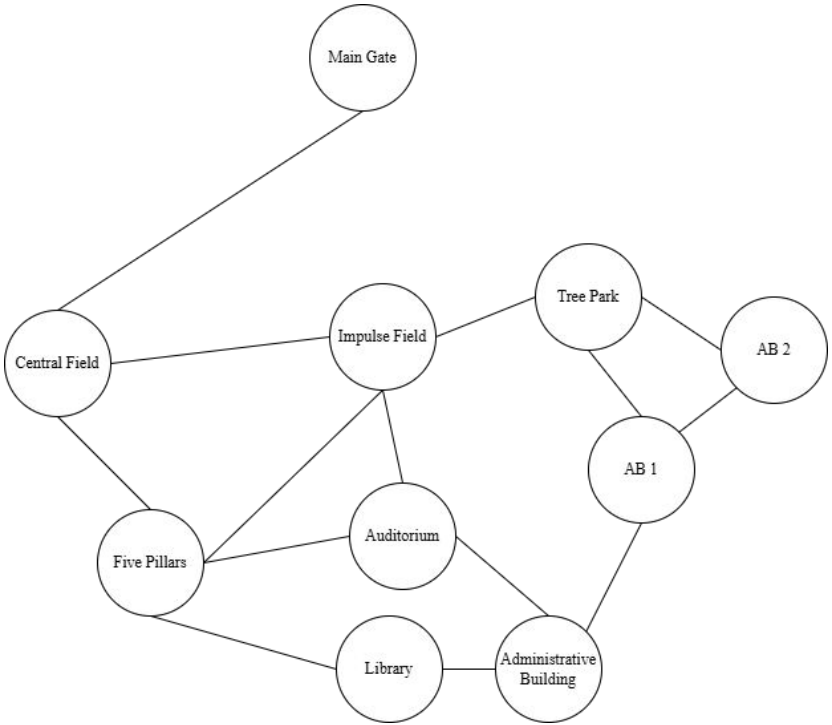
- Agent positions: 120
- # of food: 30
- # of ghosts: 2
- Ghost positions: 12
- Agent can face: NSEW

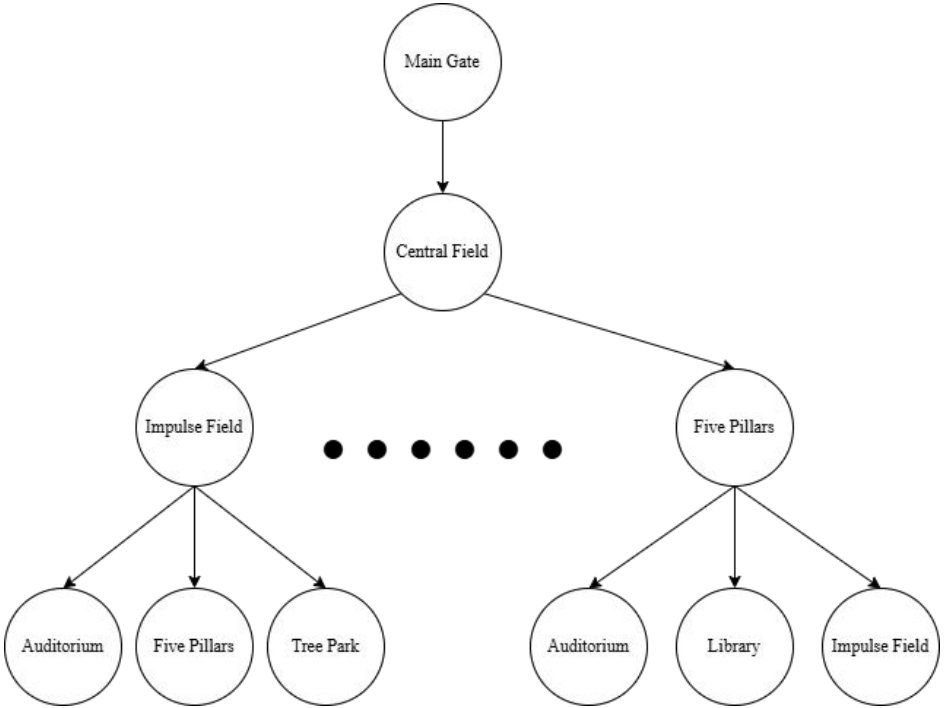
Total World states? → $120 * 2^{30} * 12^2 * 4$

Maze traversal → 120

Eat all food → $120 * 2^{30}$

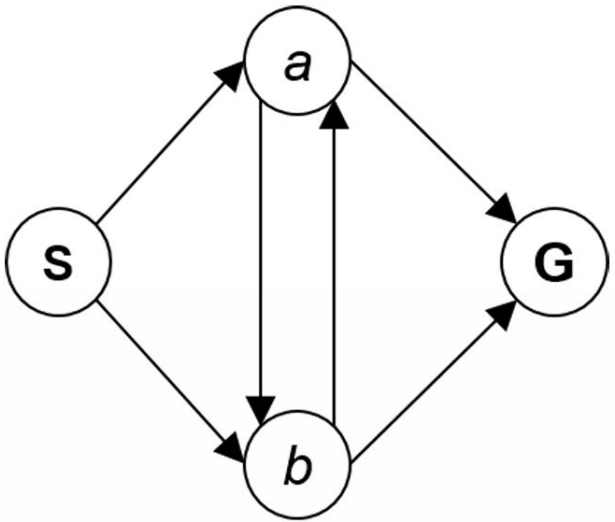
State Space Graphs and Search Trees





How big is the Search Tree?

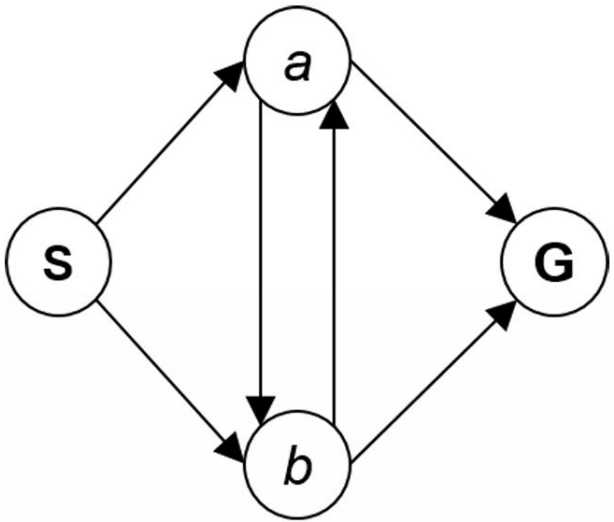
Consider this 4-state graph:



Let, s be the root.

How big is the Search Tree?

Consider this 4-state graph:

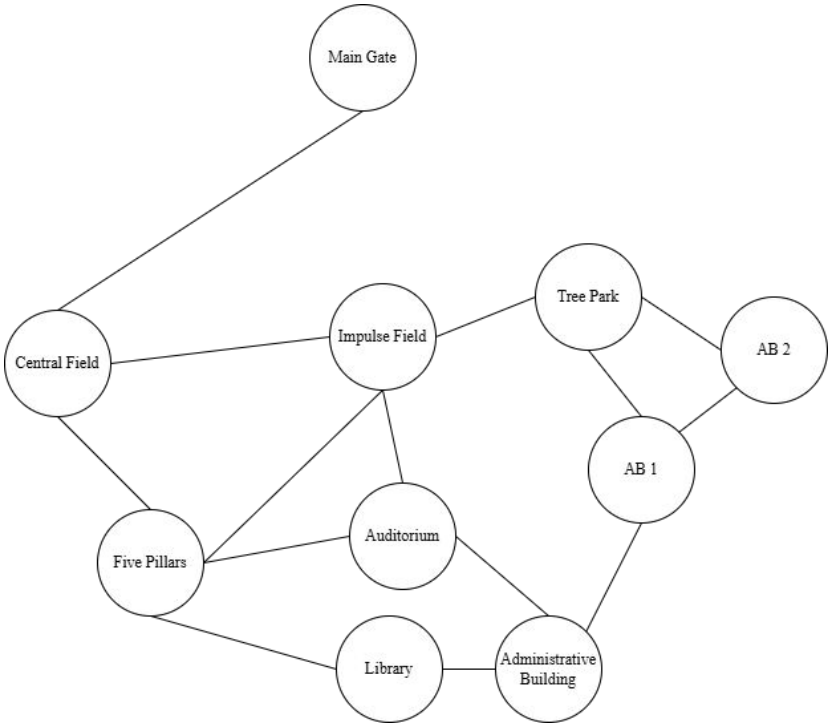


Let, s be the root.

- The search tree is infinite!
- Usually search trees have lots of repeated structures
- We can not explore the whole search tree at once

Tree Search

General Tree Search



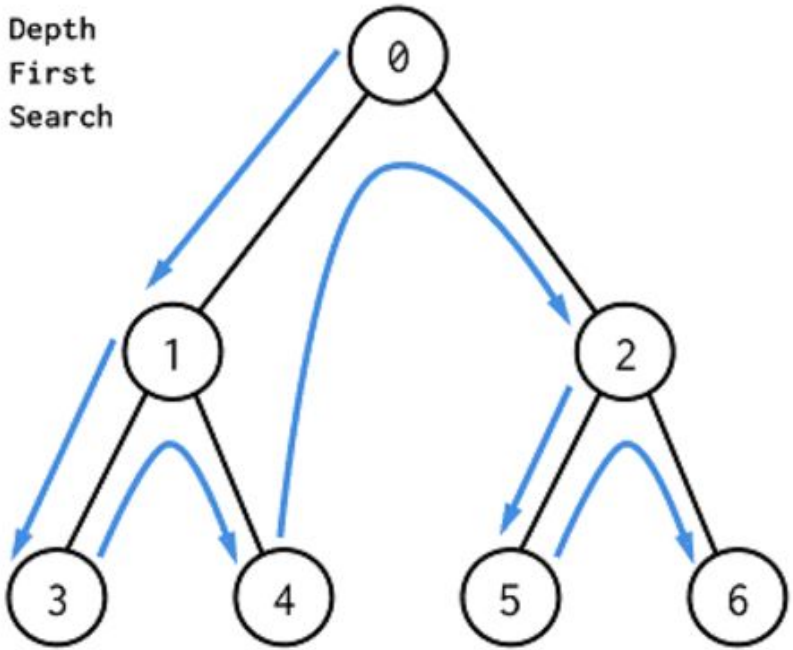
- Expand child nodes
- Maintain a list/backlog of partial routes
- Expand as few nodes as possible

General Tree Search

```
TREE-SEARCH (problem, strategy) → returns a solution or failure
    initialize tree search using the initial state of the problem
    loop do:
        if there are no candidates for expansion:
            then return failure

        choose a leaf node according to strategy
        if chosen node contains a goal state:
            then return the solution
        else:
            Expand the node & add the resulting nodes to the search tree
    end
```

Depth-First Search

Depth-First Search



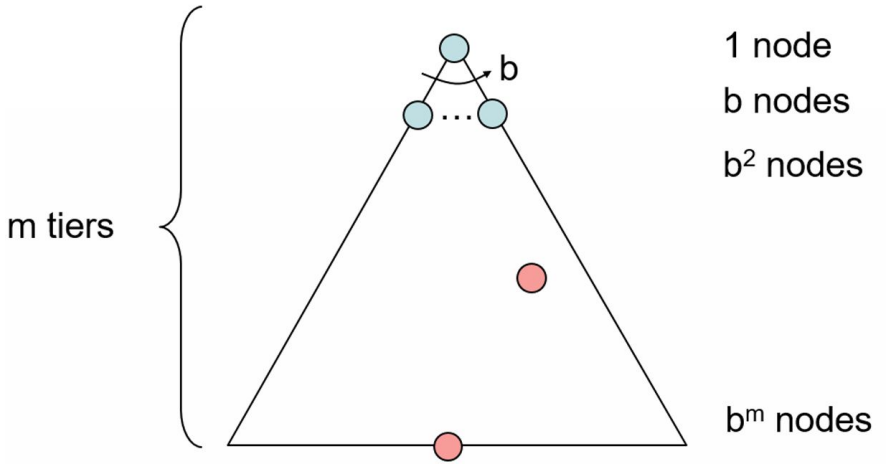
- Expand the deepest node first
- Fringe is a LIFO stack

Depth-First Search

```
DFS (problem) → returns a solution or failure
    initialize tree search using the initial state of the problem
    loop do:
        if there are no candidates for expansion:
            then return failure

        choose the most recently added leaf node
        if chosen node contains a goal state:
            then return the solution
        else:
            Expand the node & add the resulting nodes to the search tree
    end
```

Search Algorithm Properties

- Complete → Guaranteed to find a solution if it exists
- Optimal → Guaranteed to find the least cost path
- Time complexity → How many nodes does it expand?
- Space Complexity → How much space does the fringe take



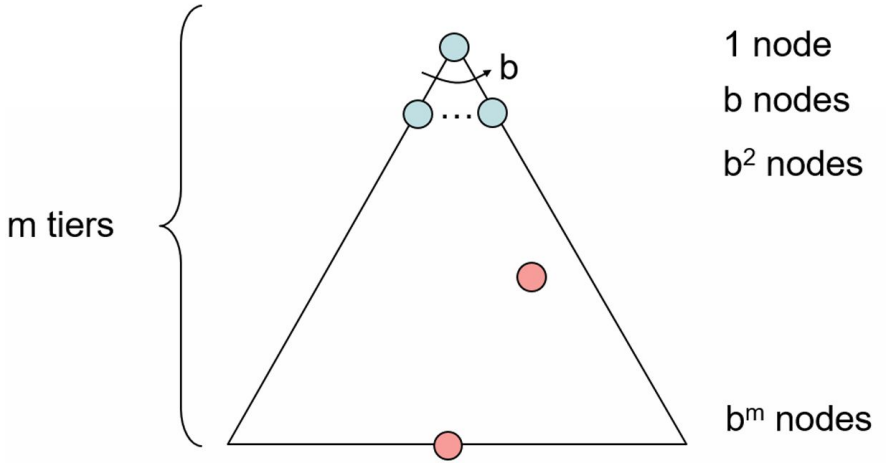
Let's consider a simplified search tree

Search Algorithm Properties

- *b* is the branching factor
- *m* is the maximum possible depth
- The red nodes are the solutions at various depths

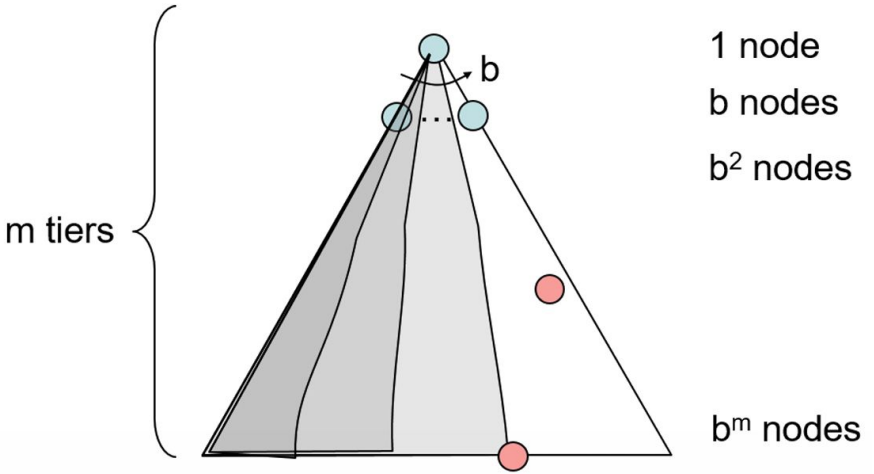
How many nodes in the whole tree?

$1 + b + b^2 + b^3 + \ldots + b^m = O(b^m)$



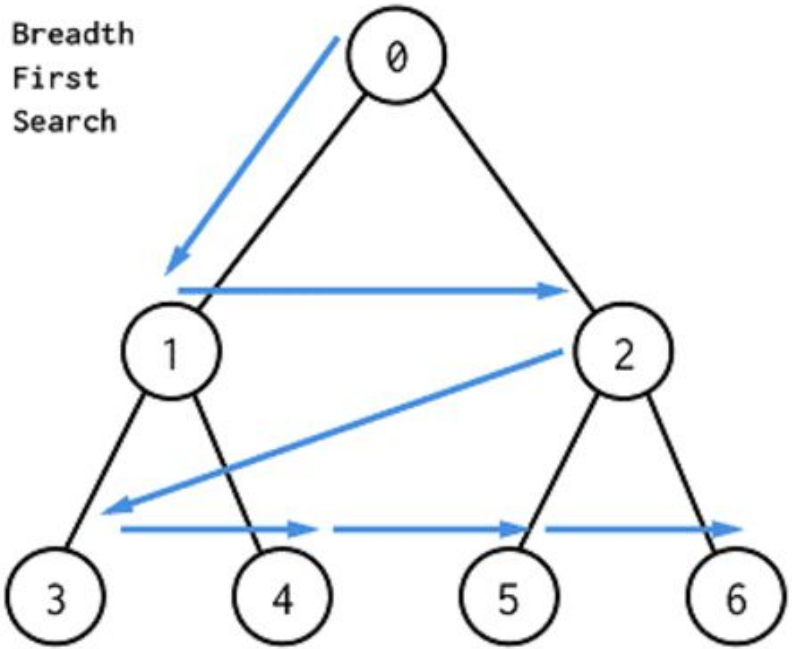
DFS Properties

- Is it complete?
  - Only if *m* is finite
- Is it optimal?
  - No, only finds the left-most solution
- Time complexity?
  - At worst case, needs to process the whole tree
  - $O(b^m)$
- Space complexity?
  - Only has siblings on the path to root
  - O(bm)



Breadth-First Search

Breadth-First Search



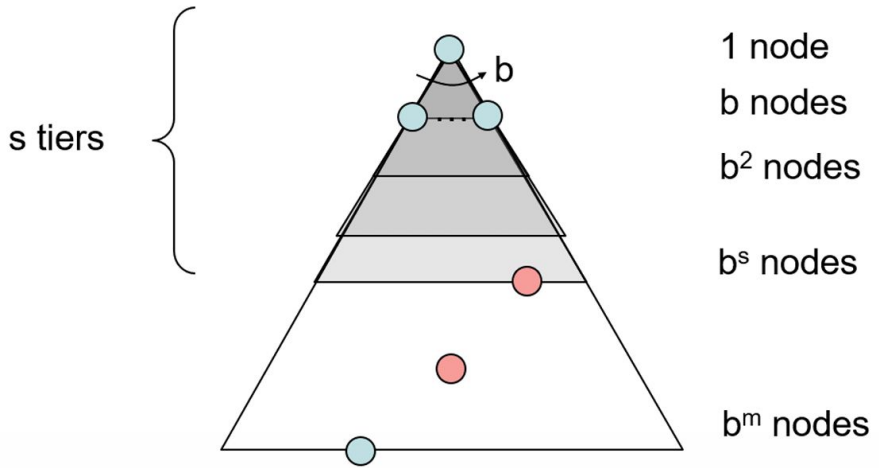
- Expand the shallowest node first
- Fringe is a FIFO queue

Breadth-First Search

```
BFS (problem) → returns a solution or failure
    initialize tree search using the initial state of the problem
    loop do:
        if there are no candidates for expansion:
            then return failure

        choose the least recently added leaf node
        if chosen node contains a goal state:
            then return the solution
        else:
            Expand the node & add the resulting nodes to the search tree
    end
```

BFS Properties

- Is it complete?
  - *s* is finite if a solution exists
- Is it optimal?
  - Only if all the costs are equal
- Time complexity?
  - Process all nodes above the shallowest solution at depth *s*
  - $O(b^s)$
- Space complexity?
  - At worst case, needs to store all the nodes at depth *s*
  - $O(b^s)$



Which one is better?

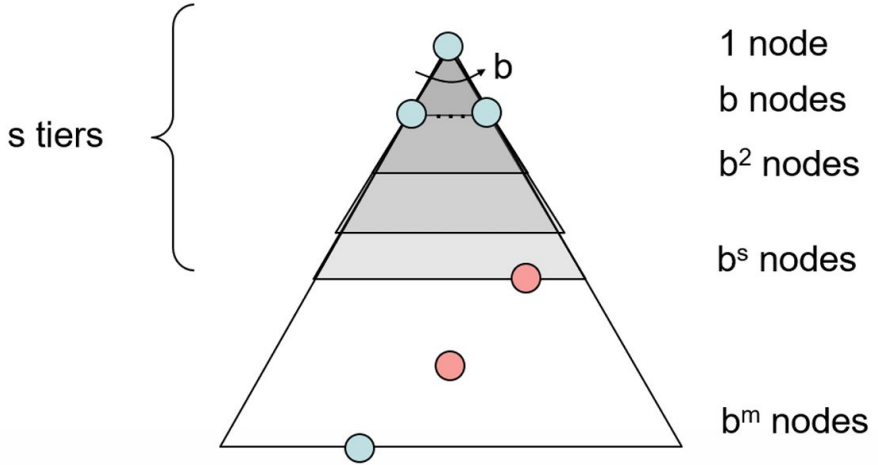
Iterative Deepening

DFS (Space advantage) + BFS (Time advantage)

- Run DFS with depth limited to 1 layer…
- Run DFS with depth limited to 2 layers…
- Run DFS with depth limited to 3 layers…
- Until shallowest solution is found!

Is it redundant? → YES 😞

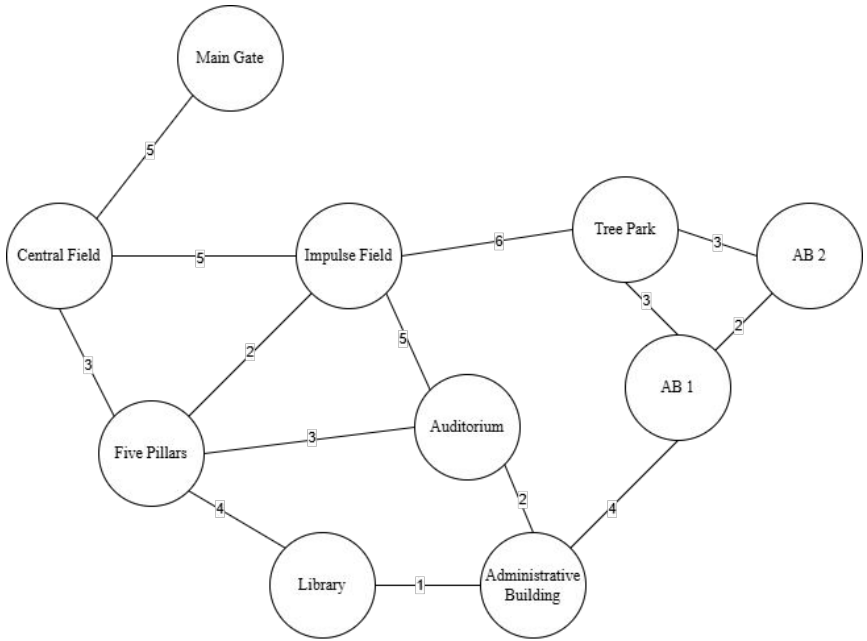
- Usually the number of nodes in the lower layers always dominates all other layers



MFW I explore the deeper layers

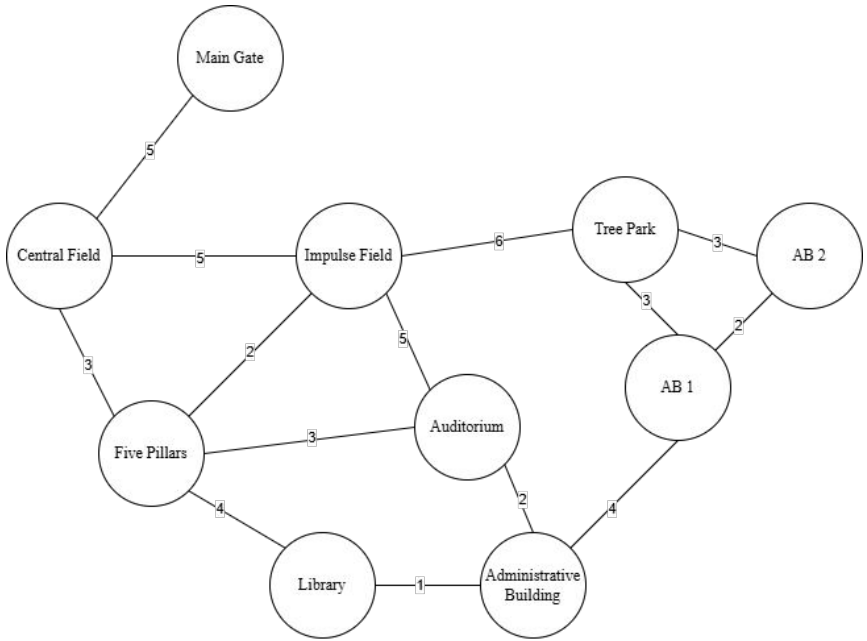
Cost Sensitive Search

Cost Sensitive Search



- What if there were some cost added to each path?
- BFS is not optimal as it won't find the least cost path, only the shallowest solution

Uniform Cost Search

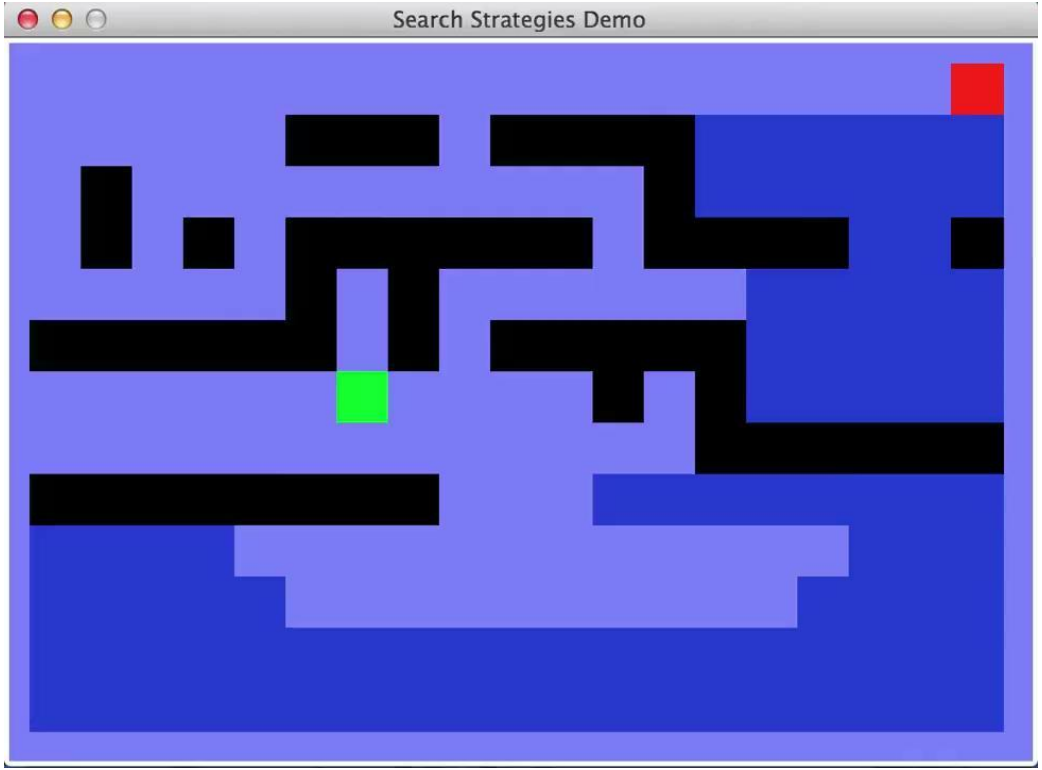


- Expand the cheapest node first
- Fringe is a priority queue
  - Priority → Cumulative Cost

Uniform Cost Search

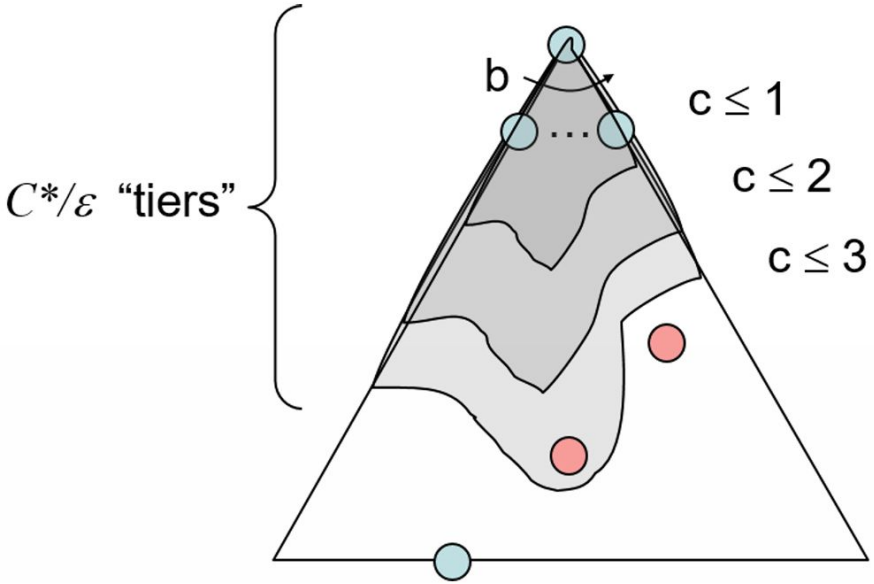
```
UCS (problem) → returns a solution or failure
    initialize tree search using the initial state of the problem
    loop do:
        if there are no candidates for expansion:
            then return failure

        choose the leaf node with lowest cumulative cost
        if chosen node contains a goal state:
            then return the solution
        else:
            Expand the node &
            add the resulting nodes and their cumulative costs
            to the search tree
    end
```

Uniform Cost Search



UCS Properties

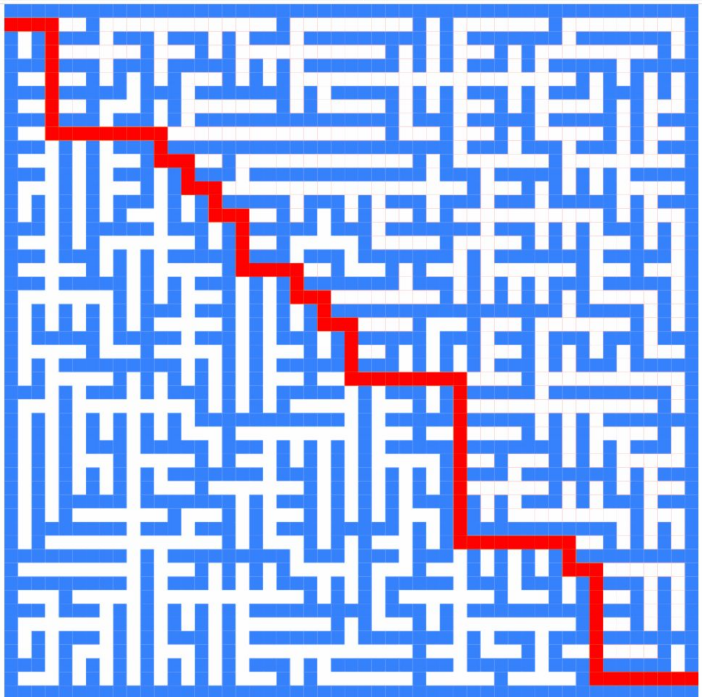
- Is it complete?
  - Only if best solution has a finite cost & the minimum cost is positive
- Is it optimal?
  - Yes. But why?
- Time complexity?
  - Process all nodes with cost less than the cheapest solution
  - If optimal solution costs C^* and each edge costs at least ε
  - Effective depth $\rightarrow O(b^{C^*/\varepsilon})$
- Space complexity?
  - $O(b^{C^*/\varepsilon})$



UCS Issues

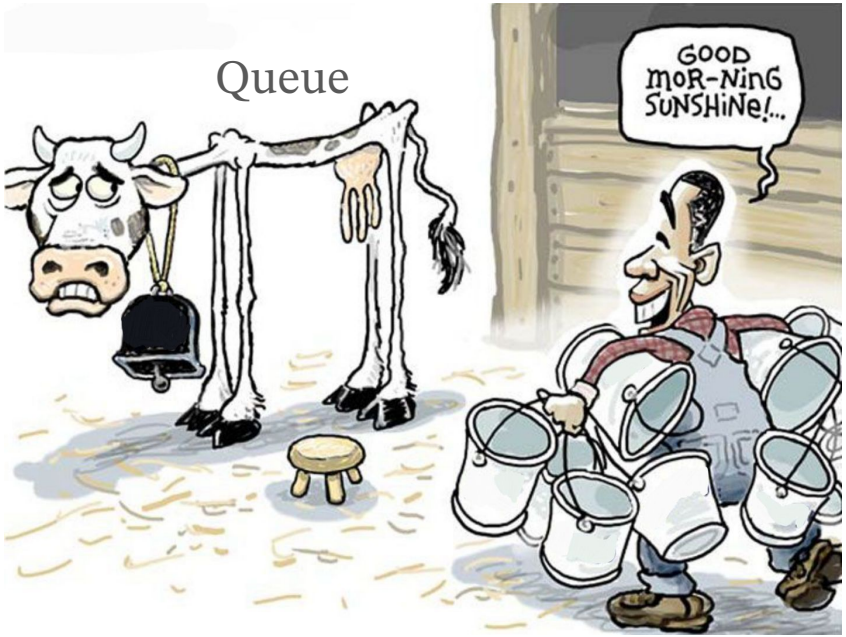
- UCS explores in every “direction” as long as the costs are lower
- No information about where the goal state is located

How could we infuse the idea of where the goal is located?



Uninformed Search Algorithms

- DFS, BFS and UCS are all the same except for the fringe strategies
- Can implement any of these algorithms just by changing the priority function in a priority queue



Additional Resources

- Maze Solving- Computerphile
- Theseus and the Minotaur | Exploring State Space
- Monte-Carlo Tree Search

Thank you