An abstract graphic on the left side of the slide, featuring a sphere-like shape composed of a dense grid of intersecting red, green, and blue lines. The lines are curved and follow the contour of the sphere, creating a complex, woven pattern. The sphere is set against a dark gray background.

10-607 Computational Foundations for Machine Learning

Data Structures

Instructor: Pat Virtue

Plan

Data Structures

- Dense vs sparse structures
- Trees
- BFS and DFS
- Stacks and Queues
- Other tree implementations
- Graphs

Abstractions vs. Data Structures

Abstractions

- List
- Set
- Map (Dictionary)
- Tree
- Queue (FIFO)
- Stack (LIFO)
- Priority Queue
- Graph

Data Structures

- Array (fixed size)
- Array (variable size)
- Linked List
- Doubly-Linked List
- Multidimensional Array
- Tensor
- Hash Map
- Binary Search Tree
- Balanced Tree
- Trie
- Stack
- Heap
- Graph
- Bipartite Graph
- Sparse Vector
- Sparse Matrix

Data Structures for ML

Examples...

Data:

- Dense feature vector (array)
- Sparse feature vector (sparse vector)
- Design matrix (multidimensional array)

Models:

- Decision Trees (tree)
- Bayesian Network (directed acyclic graph)
- Factor Graph (bipartite graph)

Algorithms:

- Greedy Search (weighted graph)
- A* Search (priority queue/heap)
- Forward-backward for HMM (trellis)

Tree to Predict C-Section Risk

Learned from medical records of 1000 women (Sims et al., 2000)

Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .05-
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .22-
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

Plan

Data Structures

- Dense vs sparse structures
- Trees
- BFS and DFS
- Stacks and Queues
- Other tree implementations
- Graphs

Dense vs sparse structures

Vector dot product example

- Linear regression model $y = \mathbf{w}^T \mathbf{x} + b$
- Where \mathbf{x} represents the contents of text data, e.g., e-mail or book review

Dense vs sparse structures

Matrix multiplication with special structure

- Example: diagonal matrices

Plan

Data Structures

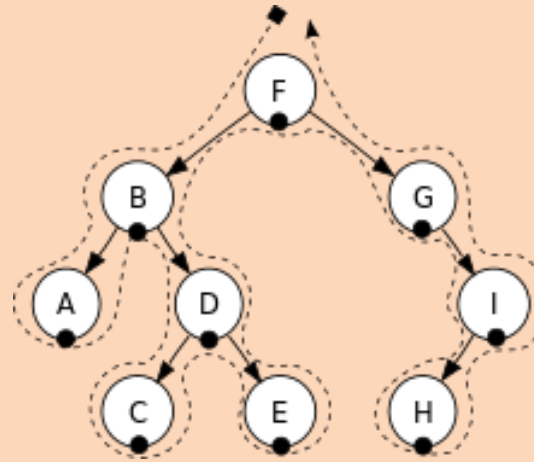
- Dense vs sparse structures
- **Trees**
- BFS and DFS
- Stacks and Queues
- Other tree implementations
- Graphs

Trees

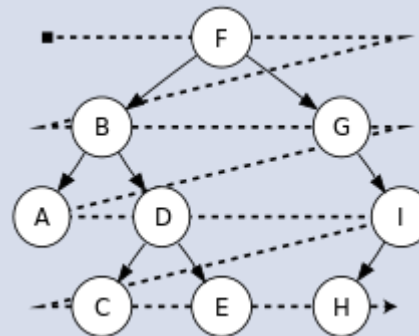
Notebook

Tree Traversals

Depth First Search

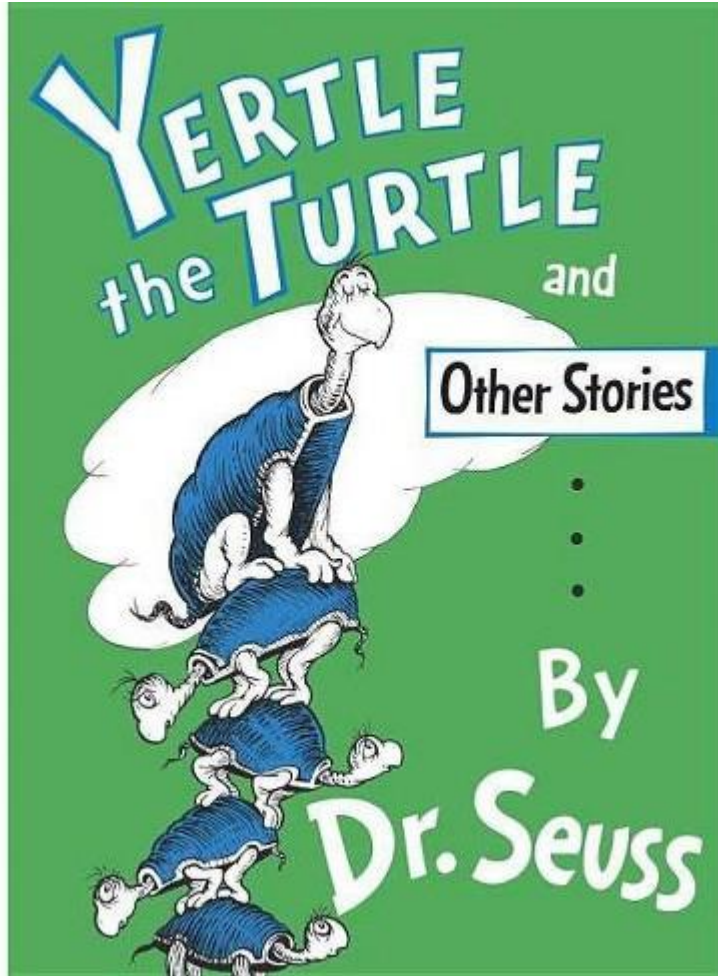


Breadth First Search



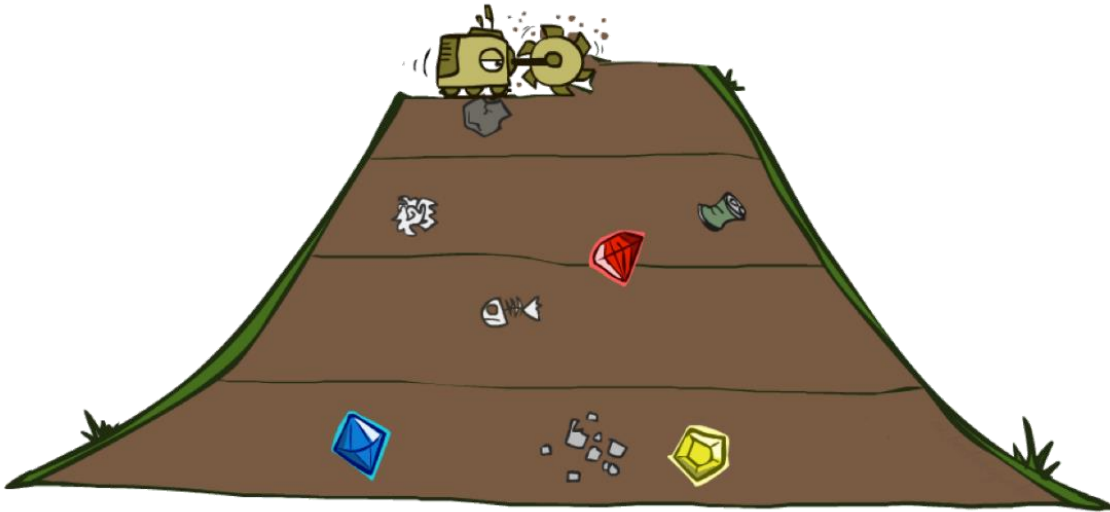
Stacks and Queues

LIFO vs FIFO

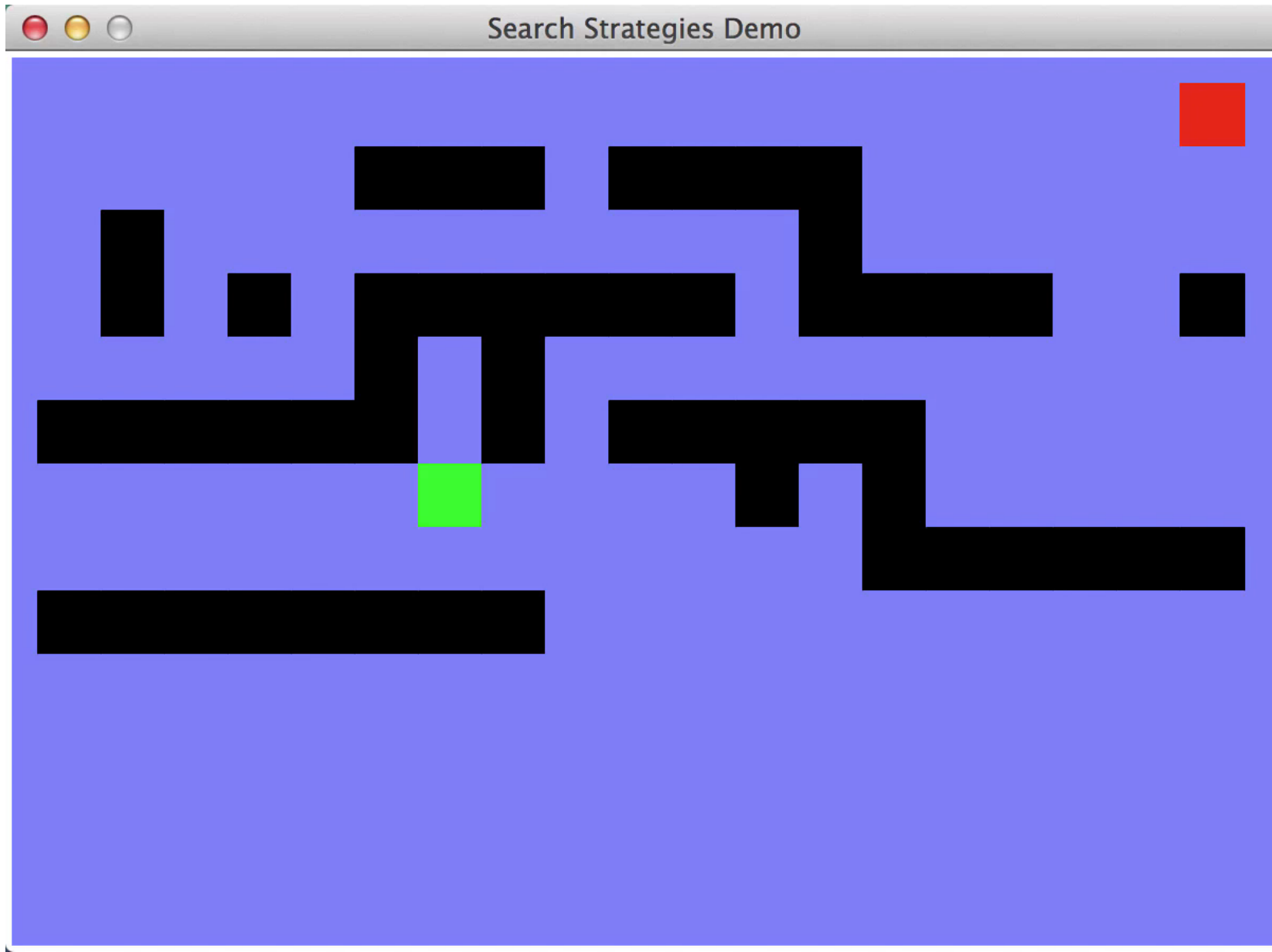


Poll 1

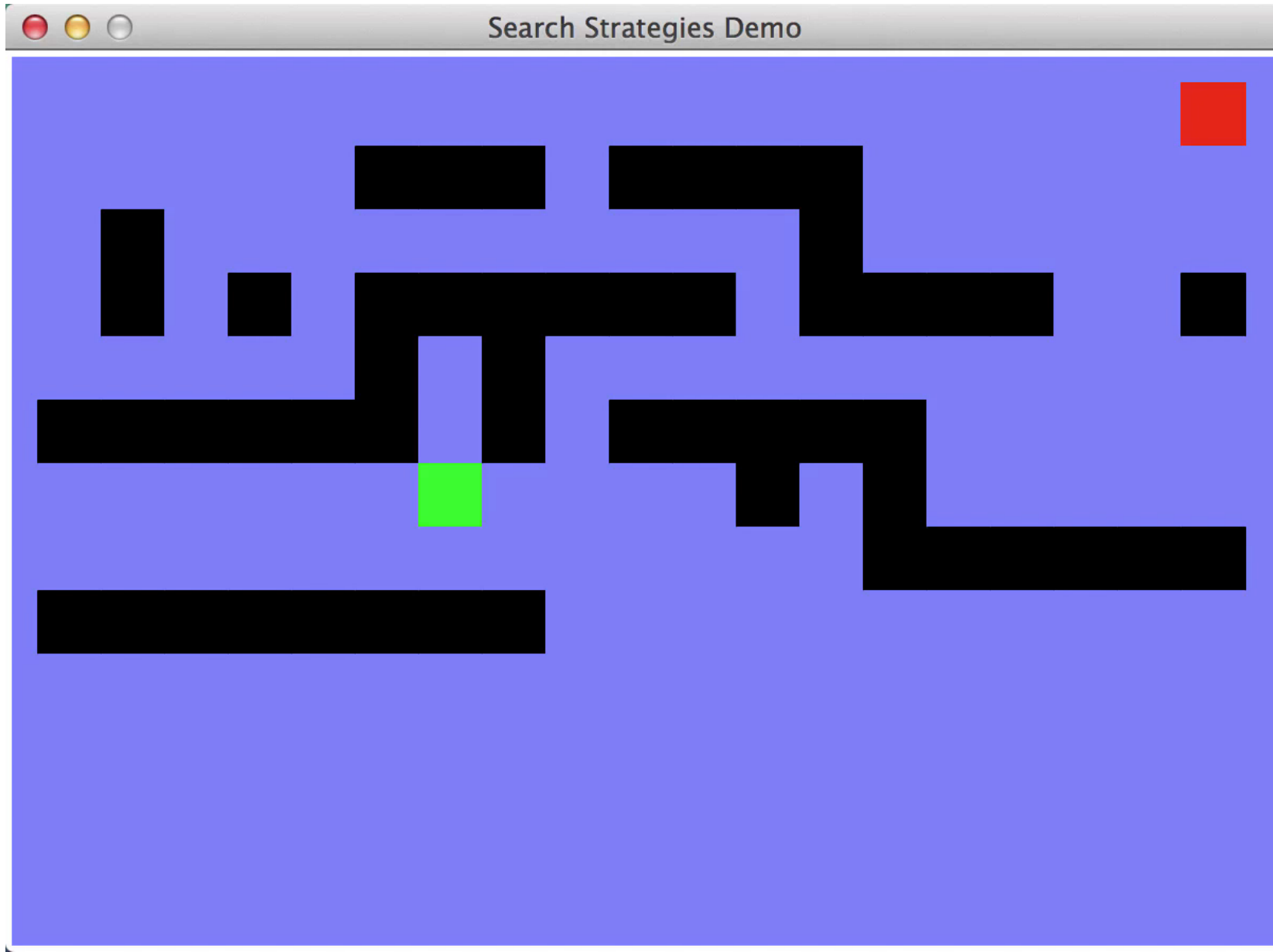
Is the following demo using BFS or DFS



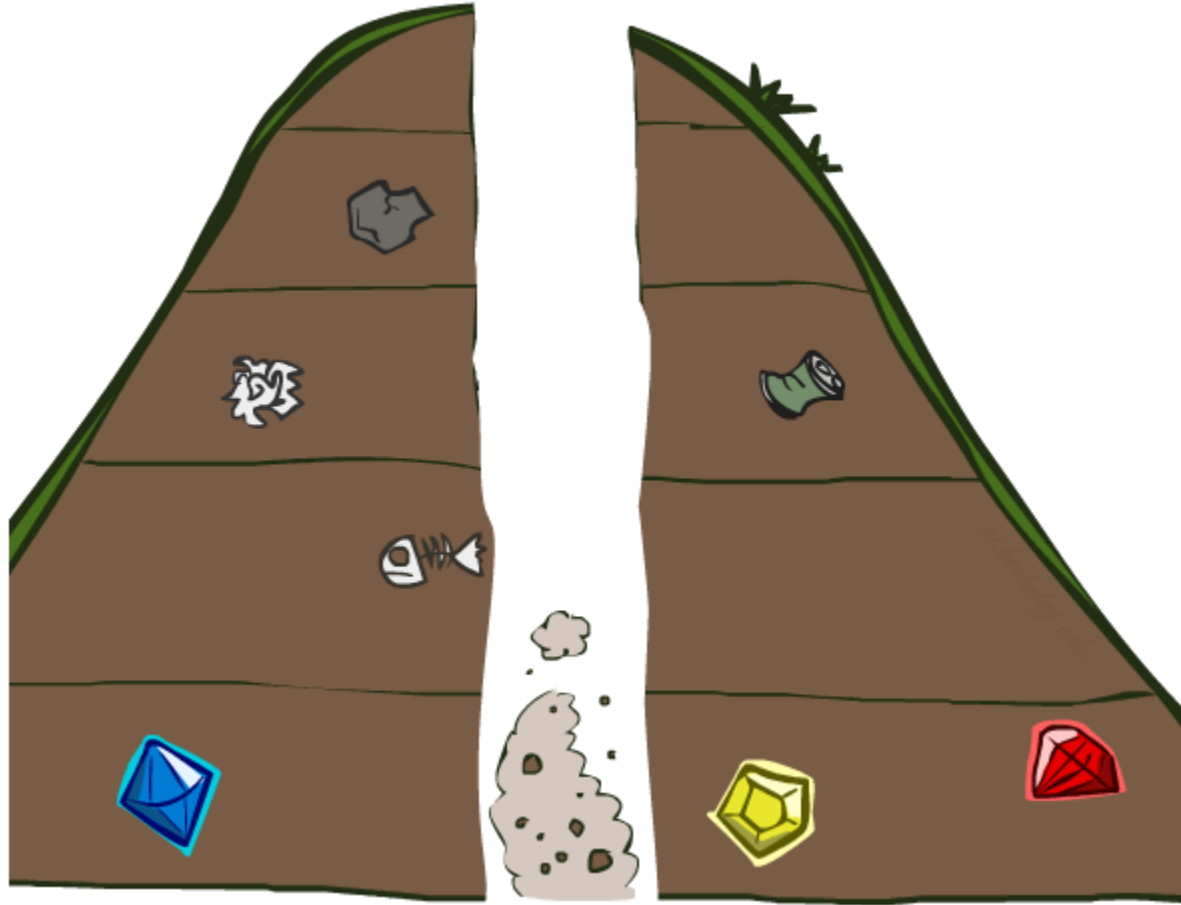
Video of Demo Maze Water DFS/BFS (part 1)



Video of Demo Maze Water DFS/BFS (part 2)



Search Algorithm Properties



Search Algorithm Properties

Complete: Guaranteed to find a solution if one exists?

Optimal: Guaranteed to find the least cost path?

Time complexity?

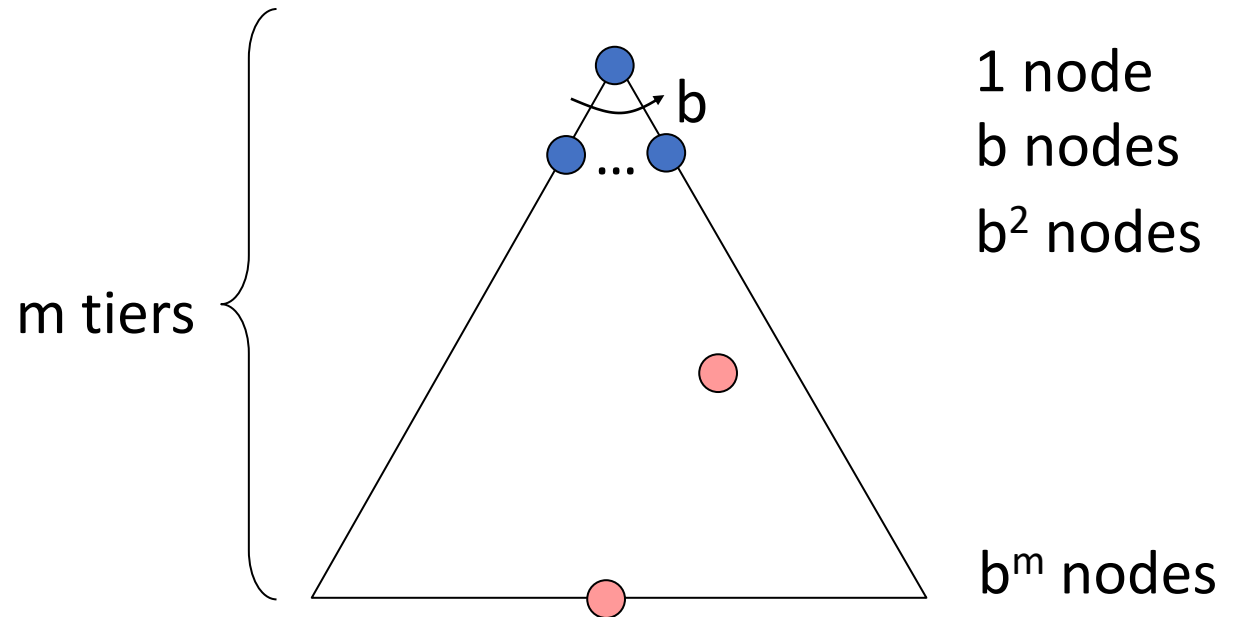
Space complexity?

Cartoon of search tree:

- b is the branching factor
- m is the maximum depth
- solutions at various depths

Number of nodes in entire tree?

- $1 + b + b^2 + \dots + b^m = O(b^{m+1})$



Search Algorithm Properties

Complete: Guaranteed to find a solution if one exists?

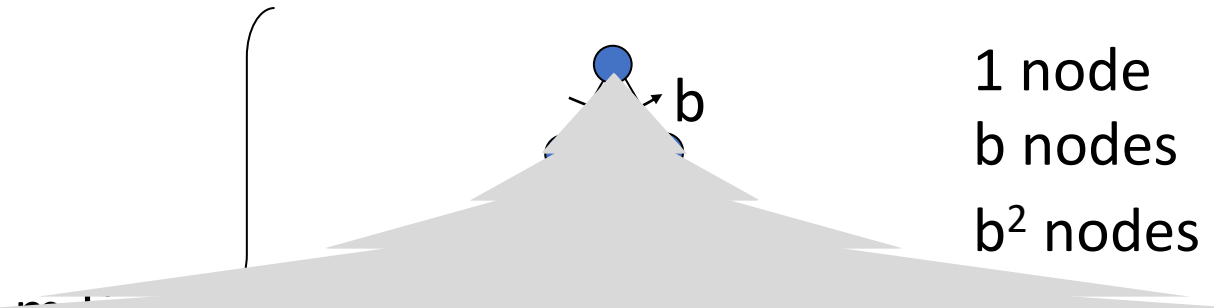
Optimal: Guaranteed to find the least cost path?

Time complexity?

Space complexity?

Cartoon of search tree:

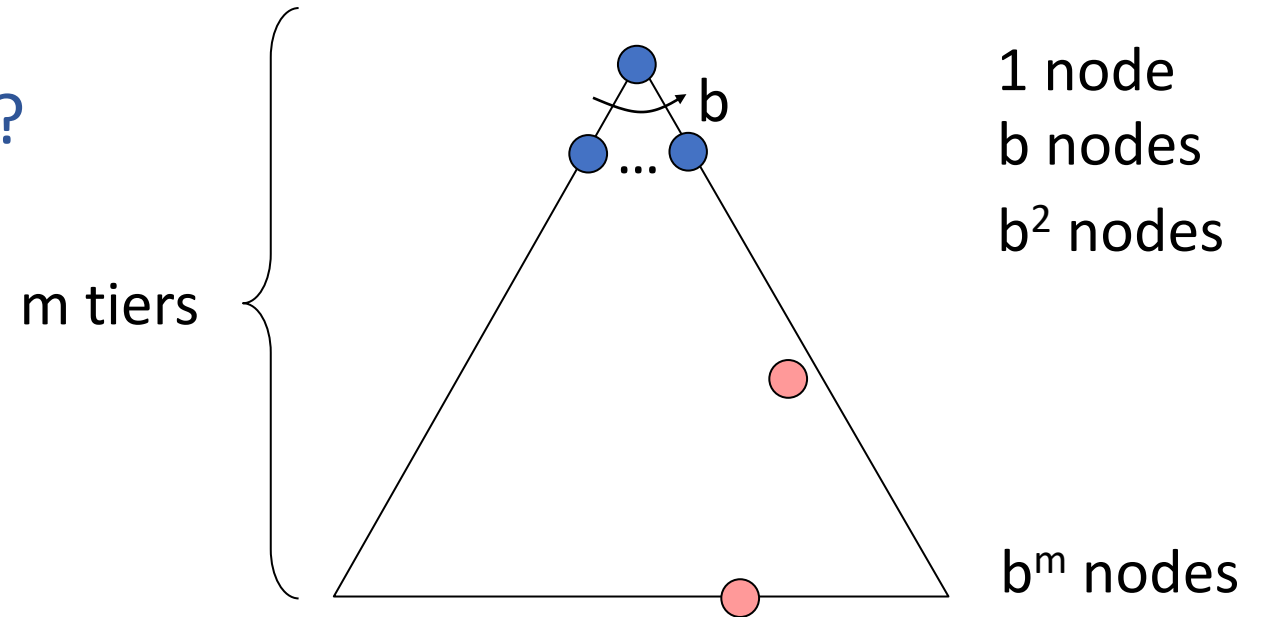
- b is the branching factor



Poll 2

Are these the properties for BFS or DFS?

- Takes $O(b^m)$ time
- Uses $O(bm)$ space on frontier
- Complete with graph search
- Not optimal unless all goals are in the same level (and the same step cost everywhere)



Depth-First Search (DFS) Properties

What nodes does DFS expand?

- Some left prefix of the tree.
- Could process the whole tree!
- If m is finite, takes time $O(b^m)$

How much space does the frontier take?

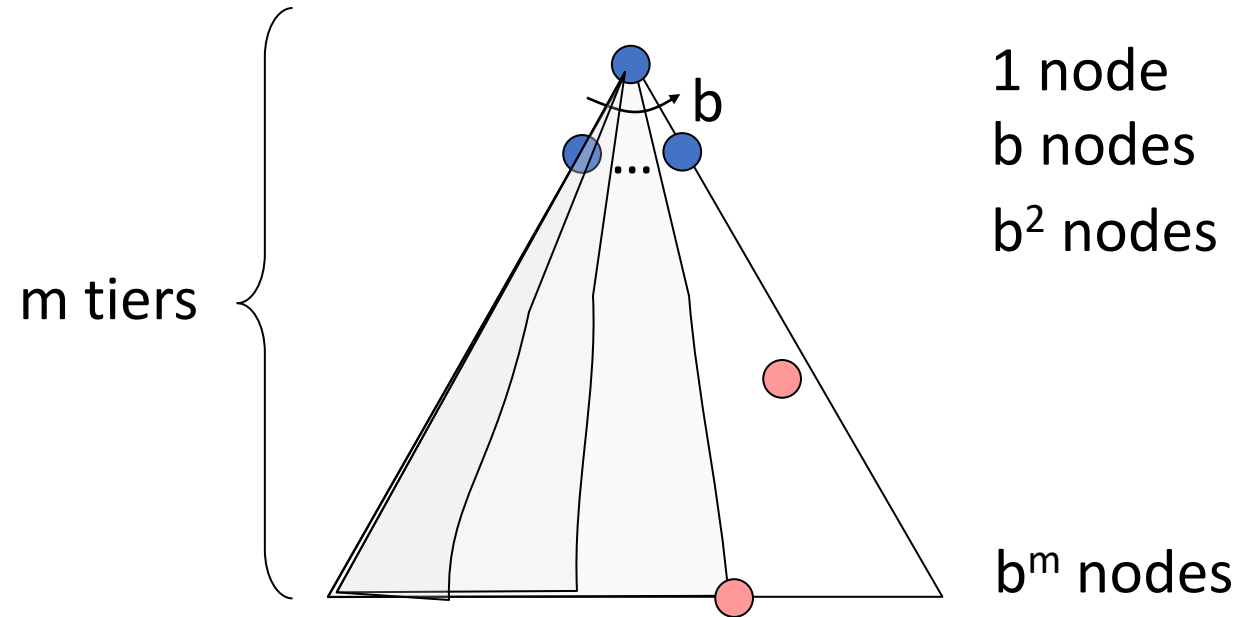
- Only has siblings on path to root, so $O(bm)$

Is it complete?

- m could be infinite, so only if we prevent cycles (graph search)

Is it optimal?

- No, it finds the “leftmost” solution, regardless of depth or cost



Breadth-First Search (BFS) Properties

What nodes does BFS expand?

- Processes all nodes above shallowest solution
- Let depth of shallowest solution be s
- Search takes time $O(b^s)$

How much space does the frontier take?

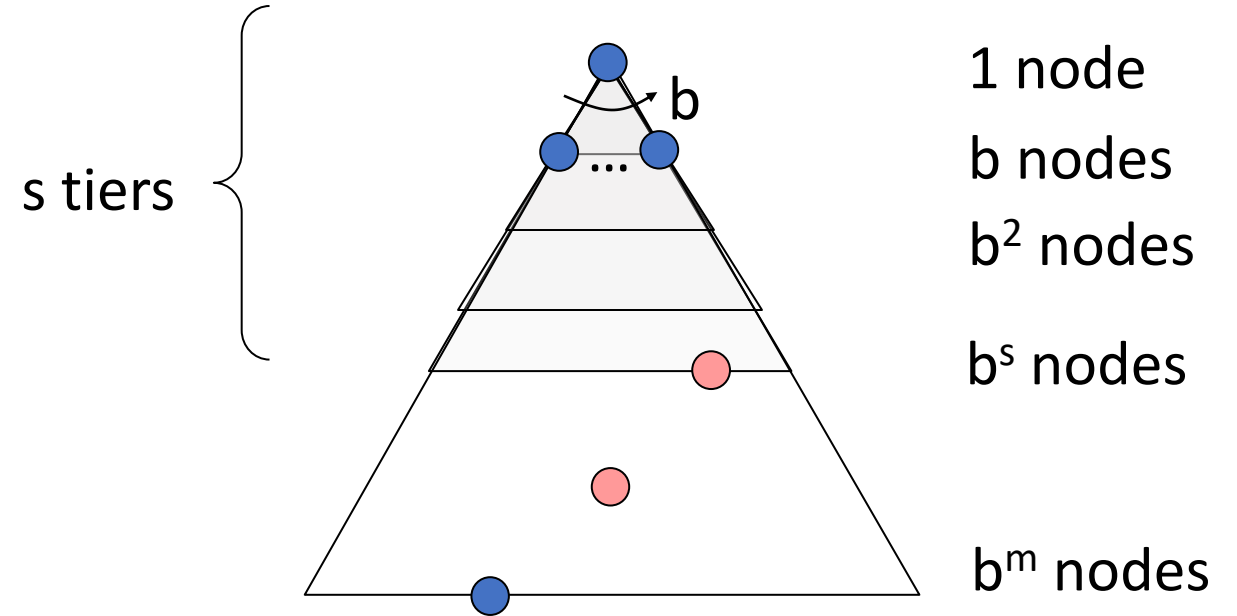
- Has roughly the last tier, so $O(b^s)$

Is it complete?

- s must be finite if a solution exists, so yes!

Is it optimal?

- Only if costs are all the same (more on costs later)



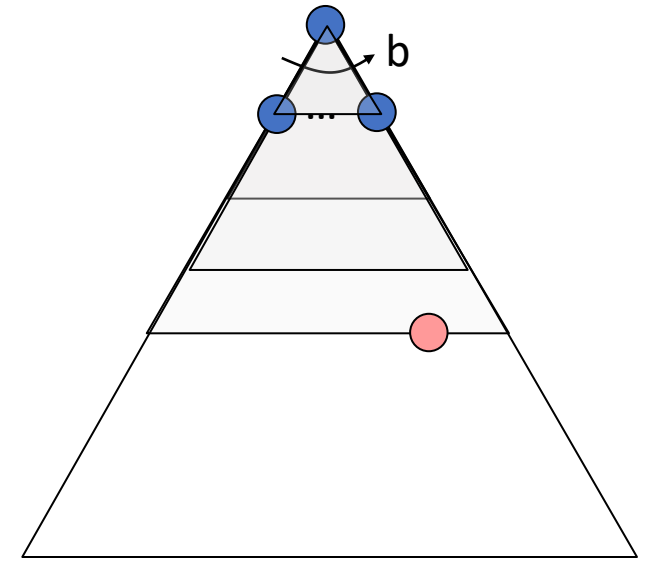
Iterative Deepening

Idea: get DFS's space advantage with BFS's time / shallow-solution advantages

- Run a DFS with depth limit 1. If no solution...
- Run a DFS with depth limit 2. If no solution...
- Run a DFS with depth limit 3.

Isn't that wastefully redundant?

- Generally most work happens in the lowest level searched, so not so bad!



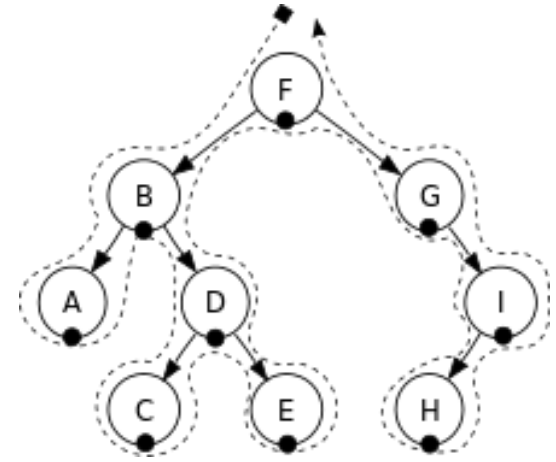
Plan

Data Structures

- Dense vs sparse structures
- Trees
- BFS and DFS
- Stacks and Queues
- **Other tree implementations**
- Graphs

Trees

Other data structures for trees



Trees

Storing information in trees

Example: Data Structure for Search (out of scope)

Nodes have

state, parent, action, path-cost

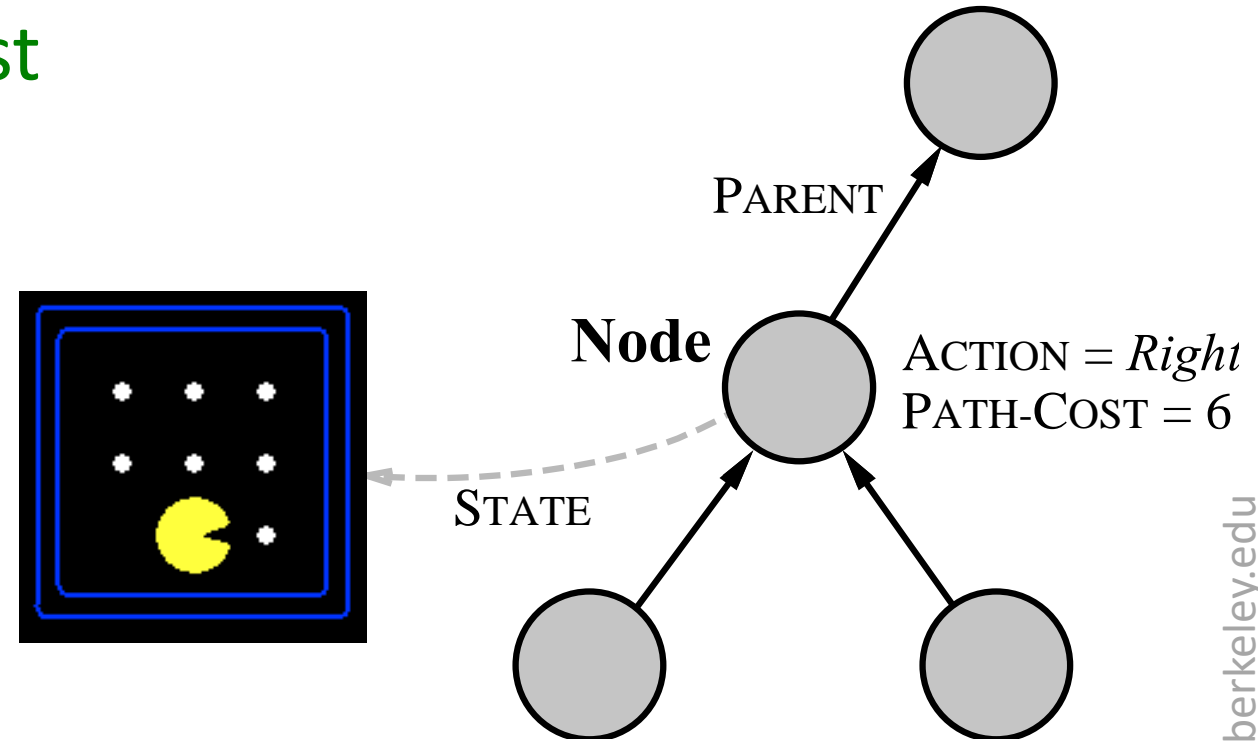
A child of node by action a has

state = result(node.state, a)

parent = node

action = a

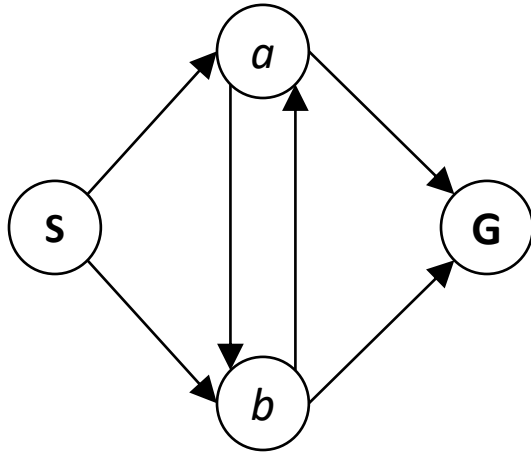
path-cost = node.path_cost +
step_cost(node.state, a , self.state)



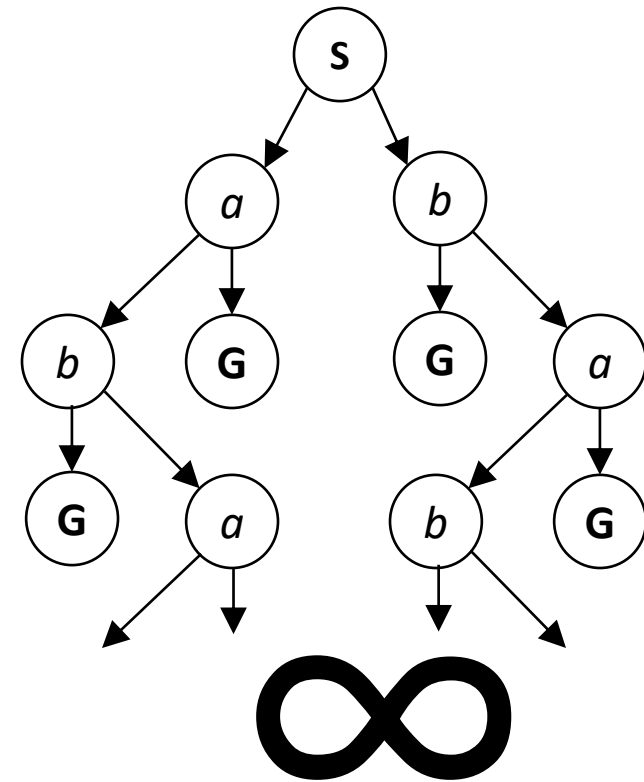
Extract solution by tracing back parent pointers, collecting actions

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!

Tree Search vs Graph Search

