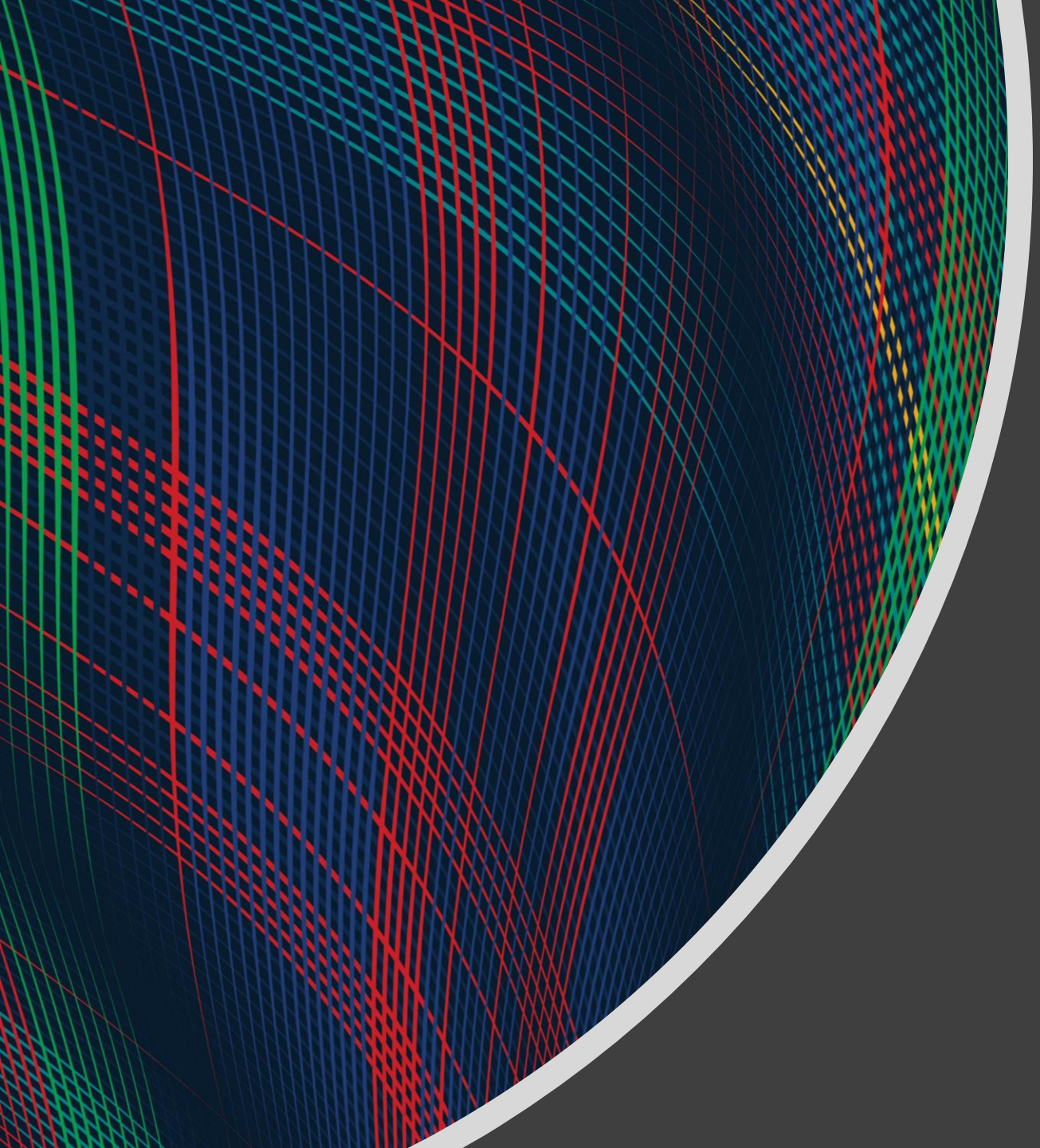


10-607 Computational Foundations for Machine Learning

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)

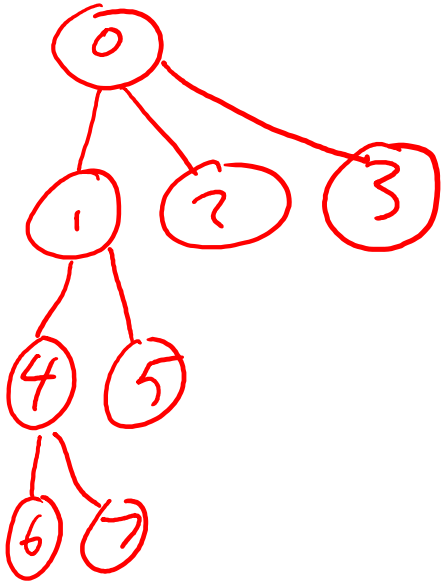
Algorithms:

- 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




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

Figure from Tom Mitchell

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

x_1 : 1 : free , 0: o.w

x_2 : # All CAPS words

x_3 : # money appears

$w_1 x_1 + w_2 x_2 + w_3 x_3 + b$

Vocabulary with $|V|$ num of words

x_1 : 1 if first word in V in text, 0 o.w.

x_2 : 1 if second

x_{1200} : 1 if cat in text, o.w.

0 0
10 1
0 0
3 1

bad

[(2, 10), (1200, 3)]

[2, 1200]
w_2 w_{1200}

Dense vs sparse structures

Matrix multiplication with special structure

- Example: diagonal matrices

Plan

Data Structures

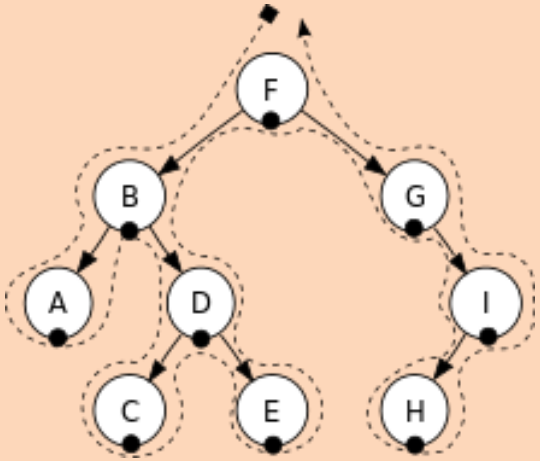
- Dense vs sparse structures
- Trees
- BFS and DFS
- Stacks and Queues
- Other tree implementations
- Graphs

Trees

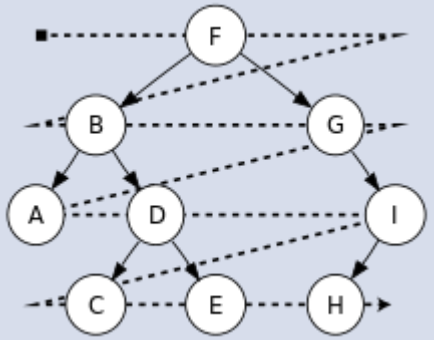
Notebook

Tree Traversals

Depth First Search



Breadth First Search

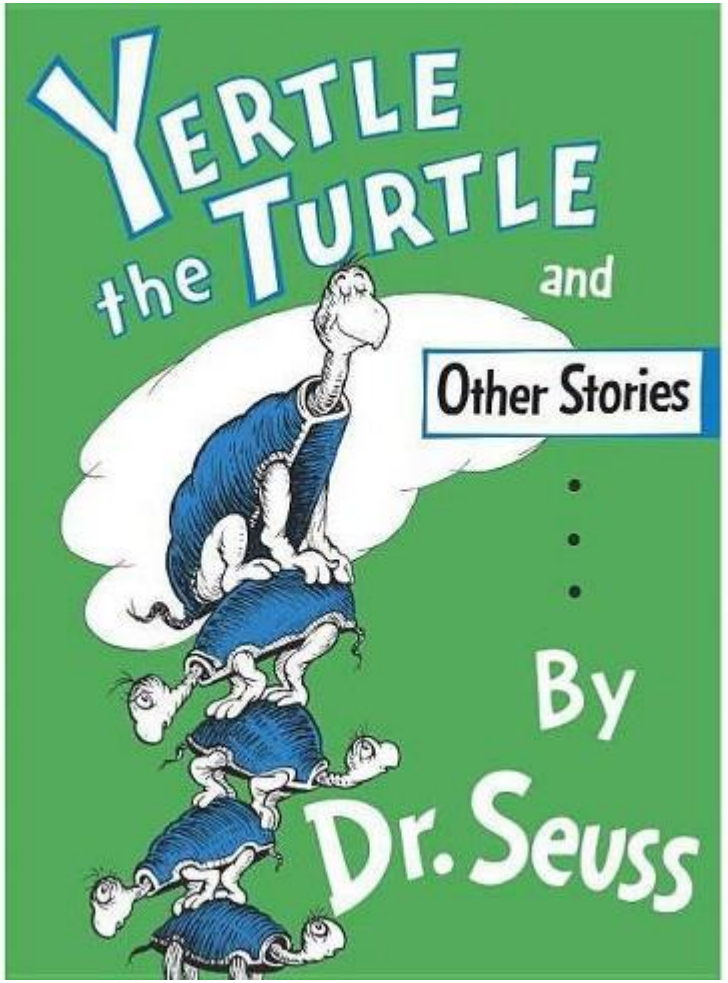


Figures from Wikipedia

Stacks and Queus

LIFO vs FIFO

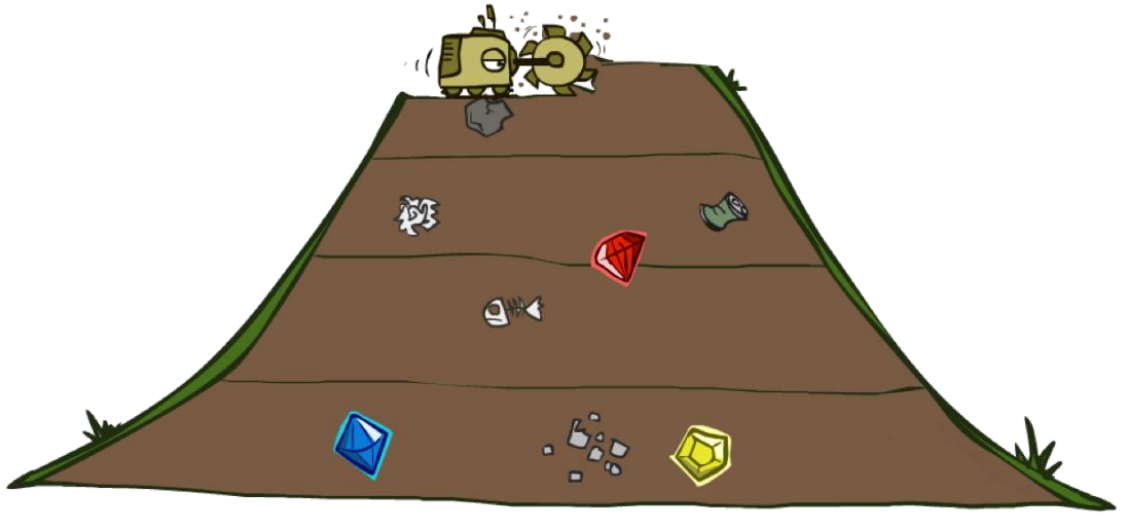
LIFO
Stack

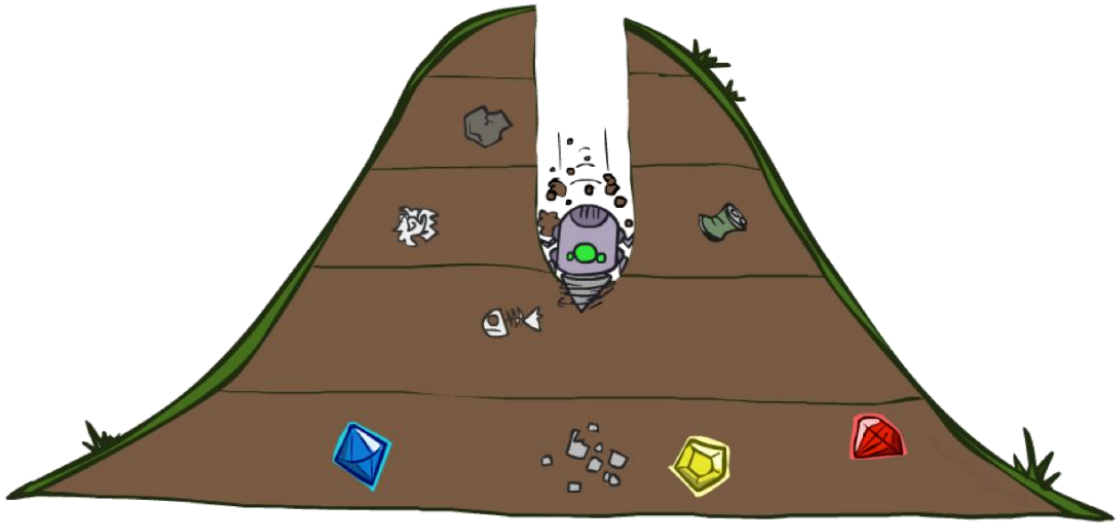


FIFO
Queue

Poll 1

Is the following demo using BFS or DFS

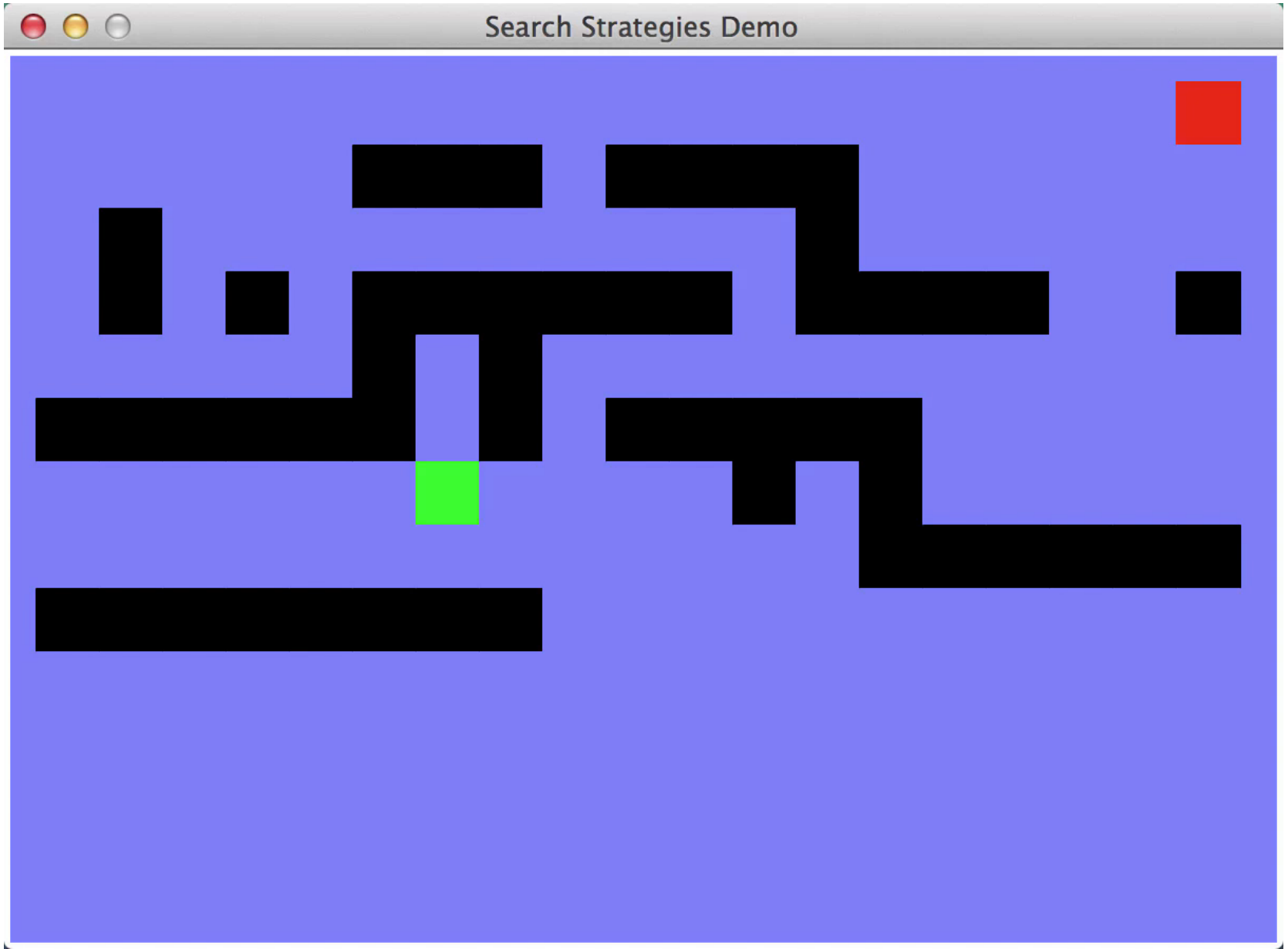




Slide credit: ai.berkeley.edu

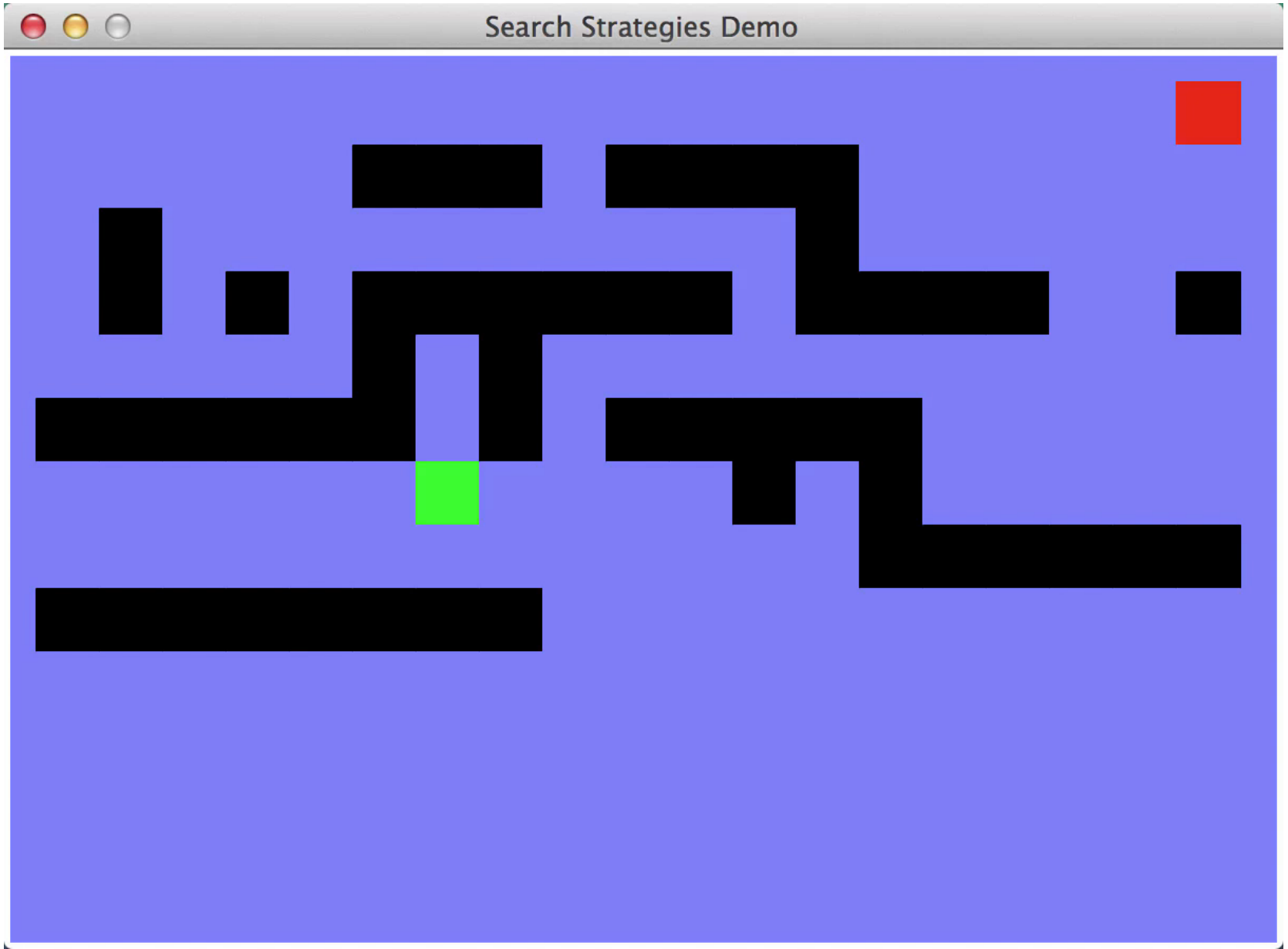
[Demo: dfs/bfs maze water (L2D6)]

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



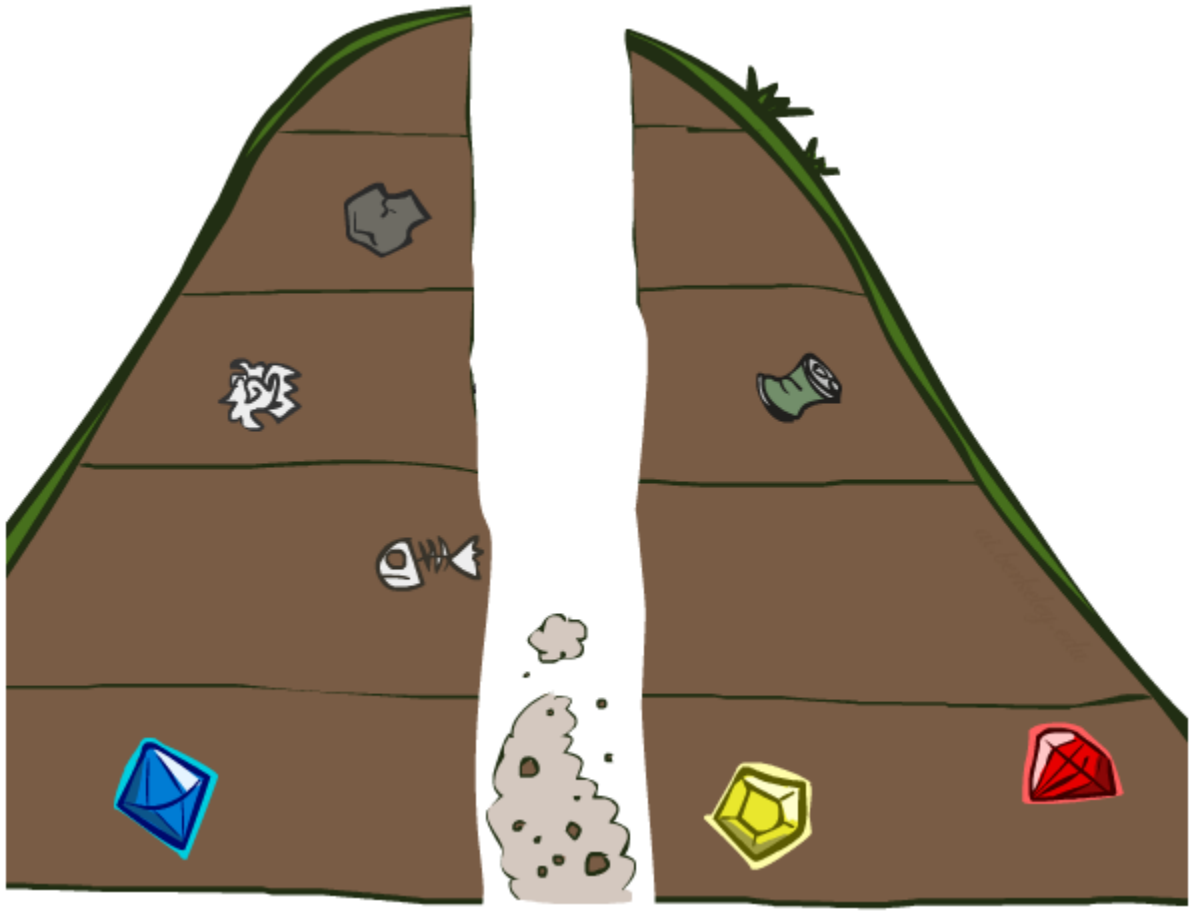
Slide credit: ai.berkeley.edu

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



Slide credit: ai.berkeley.edu

Search Algorithm Properties



Slide credit: ai.berkeley.edu

Search Algorithm Properties

Complete: Guaranteed to find a solution if one exists?

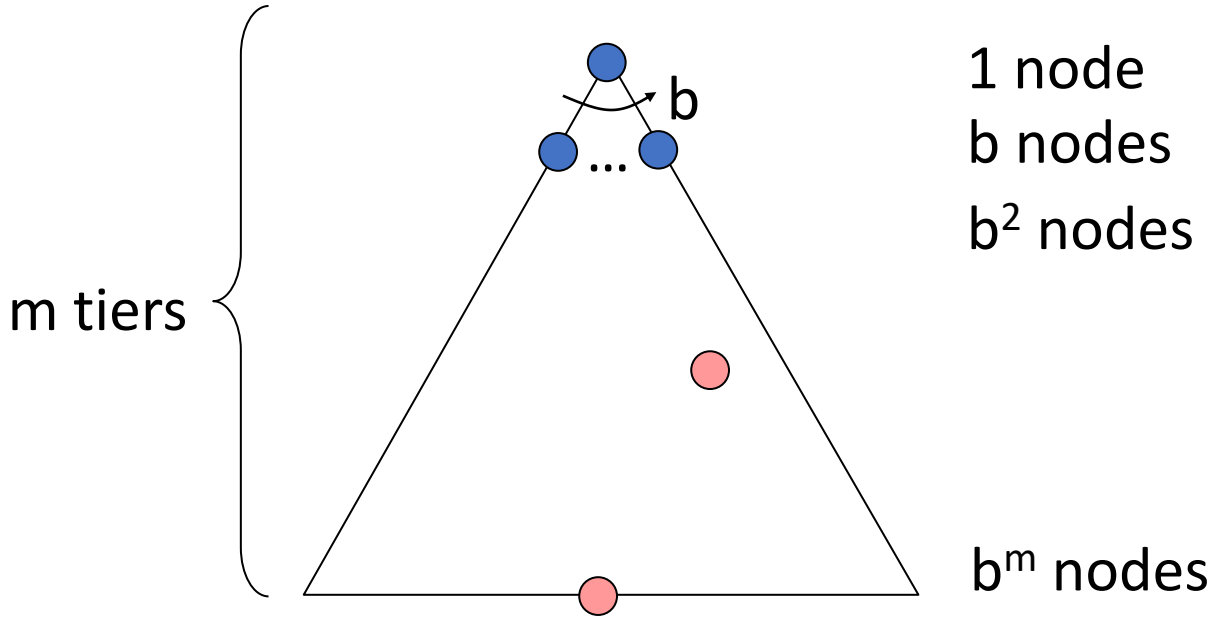
Optimal: Guaranteed to find the least cost path?

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?

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

Slide credit: ai.berkeley.edu

Search Algorithm Properties

Complete: Guaranteed to find a solution if one exists?

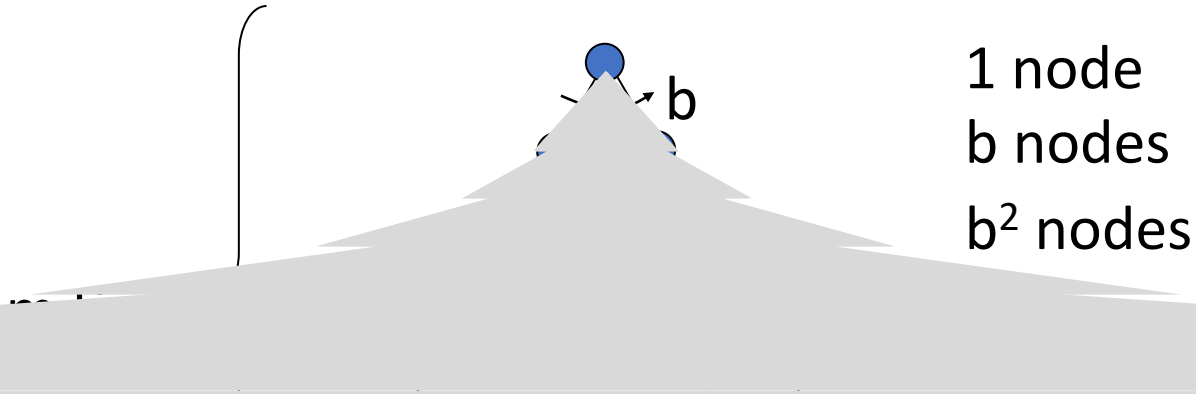
Optimal: Guaranteed to find the least cost path?

Time complexity?

Space complexity?

Cartoon of search tree:

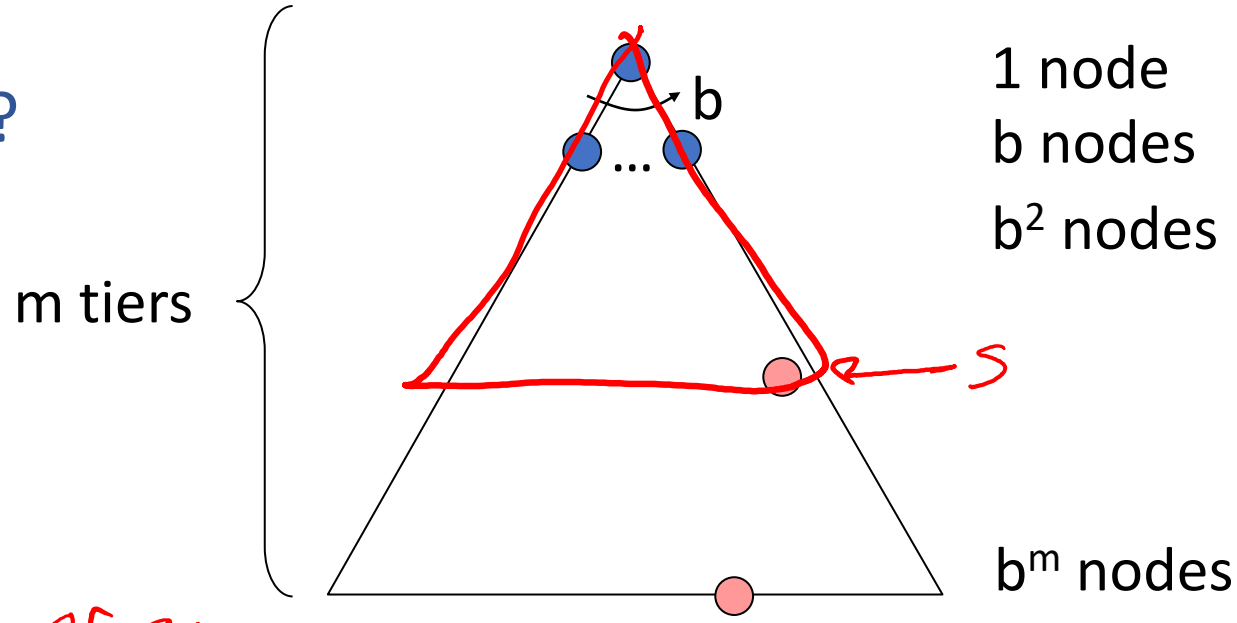
- b is the branching factor



Slide credit: ai.berkeley.edu

Poll 2

Are these the properties for BFS or DFS?



- Takes $O(b^m)$ time
- Uses O(bm) space on frontier

stack or queue

- Complete with graph search
- Not optimal unless all goals are in the same level (and the same step cost everywhere)

Slide credit: ai.berkeley.edu

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?

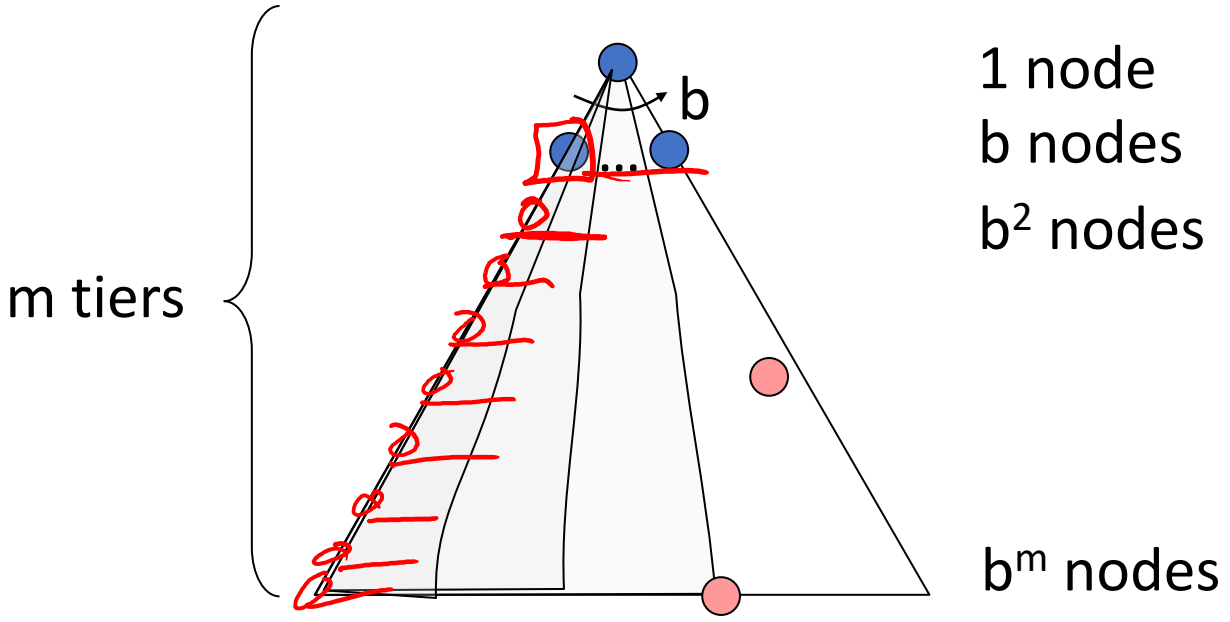
- Only has siblings on path to root, so O(bm)

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



Slide credit: ai.berkeley.edu

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?

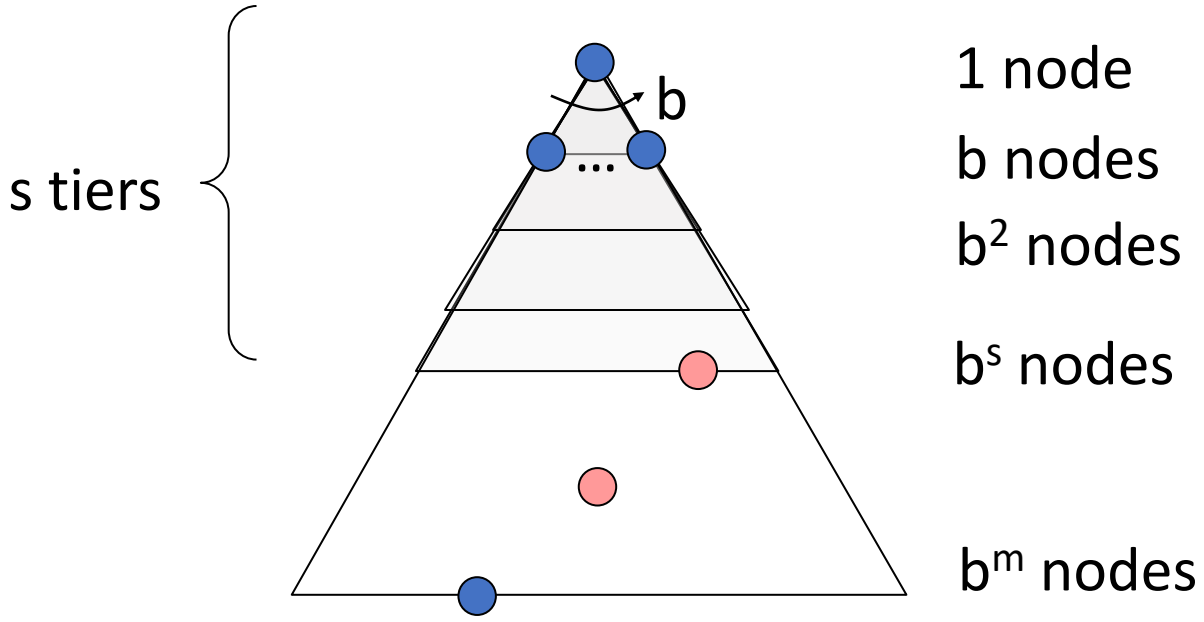
- Has roughly the last tier, so $O(b^s)$

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)

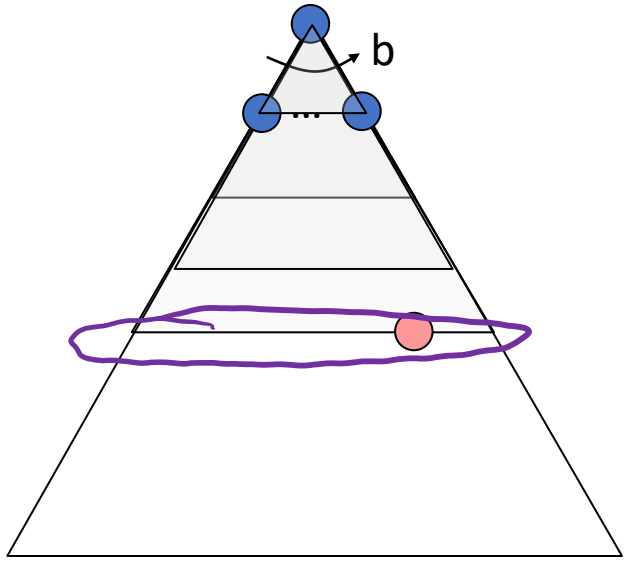


Slide credit: ai.berkeley.edu

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!

Slide credit: ai.berkeley.edu

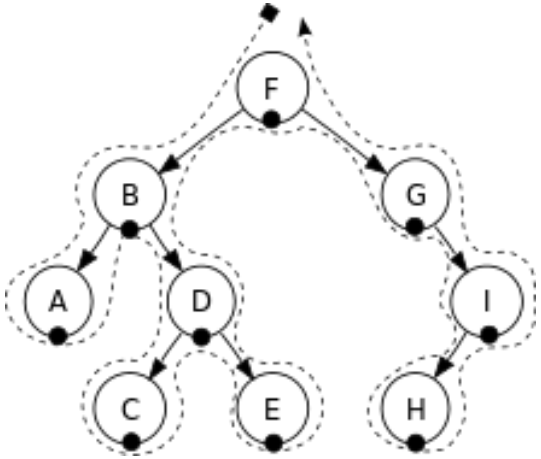
Plan

Data Structures

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

Trees

Other data structures for trees



(F, (—, —))

(D, ((C, ()), (E, ())))

Figure from Wikipedia

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

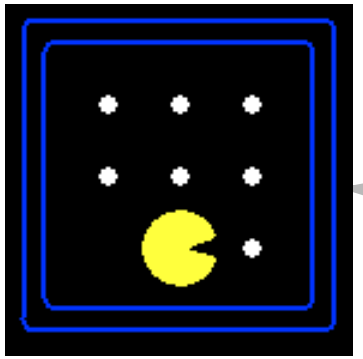
state = result(node.state,*a*)

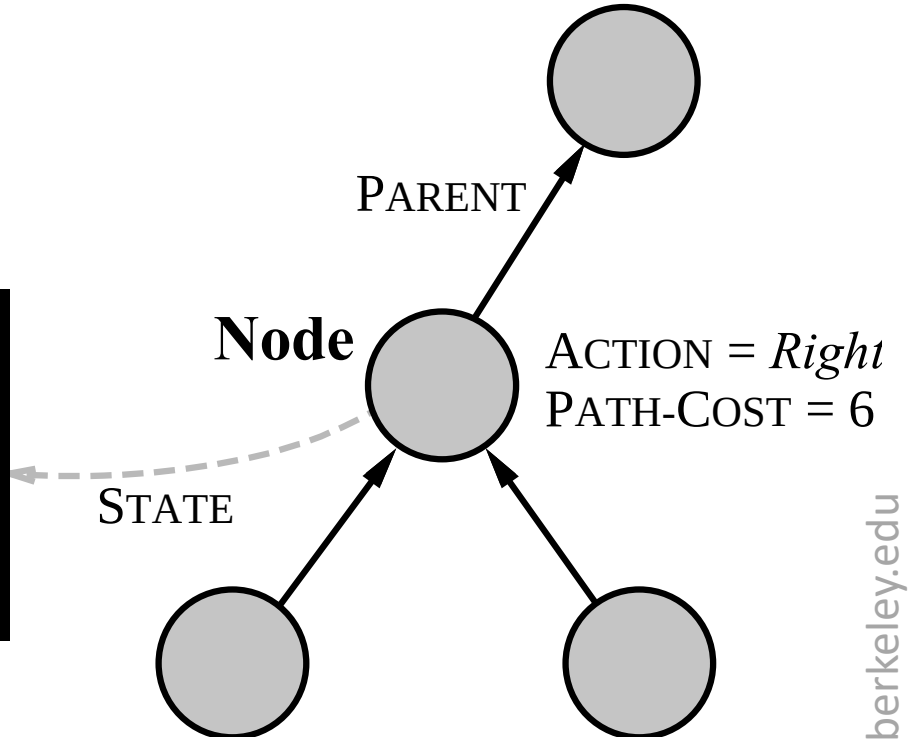
parent = node

action = *a*

path-cost = node.path_cost +

step_cost(node.state, *a*, self.state)

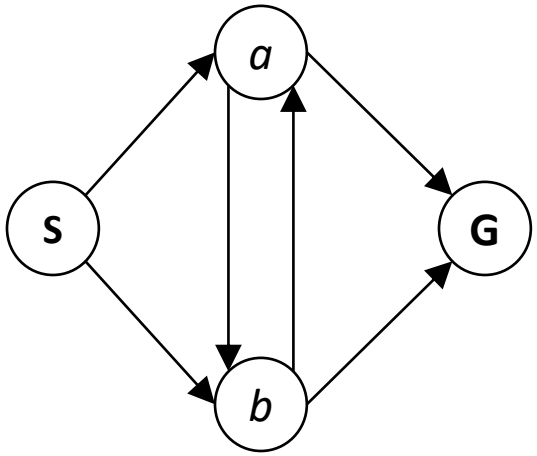




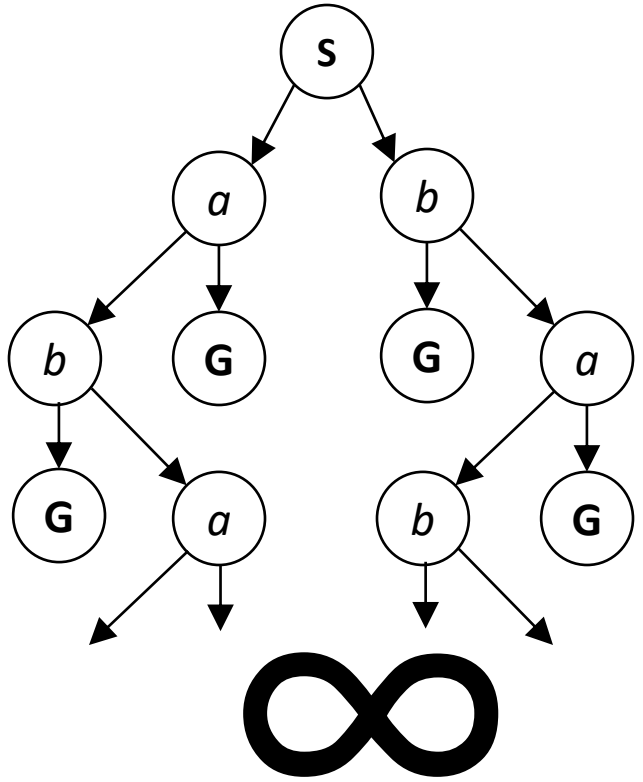
Extract solution by tracing back parent pointers, collecting actions

State Space Graphs vs. Search Trees

Consider this 4-state graph:



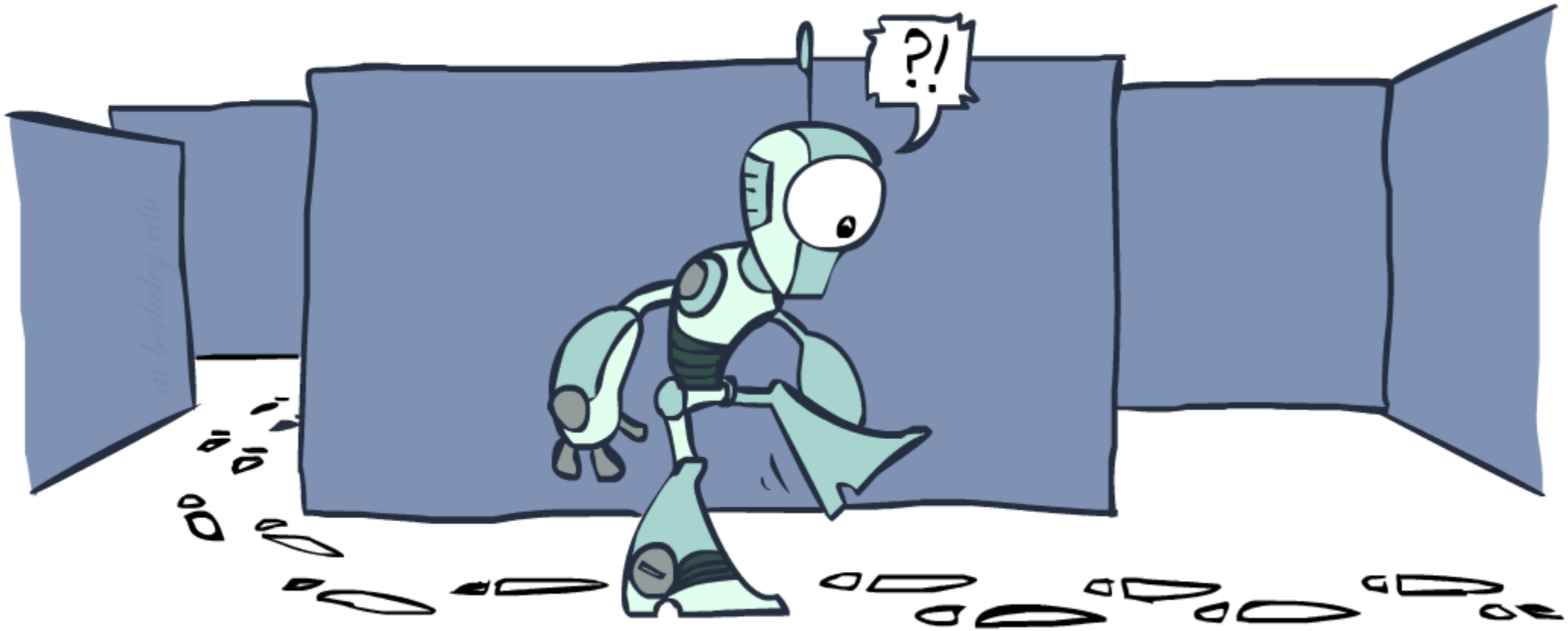
How big is its search tree (from S)?



Important: Lots of repeated structure in the search tree!

Slide credit: ai.berkeley.edu

Tree Search vs Graph Search



Slide credit: ai.berkeley.edu