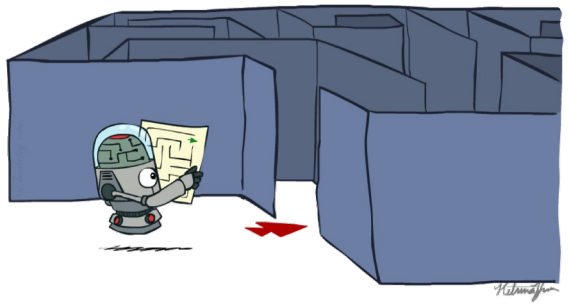
Uninformed Search

CSE 473: Introduction to Artificial Intelligence



James Weichert (jpw@cs)
cs.uw.edu/473
acknowledgements

Welcome!
Place in Washington you want to visit?
sli.do **#cse473**

Administrivia

ANNOUNCEMENTS

- Remember to complete the **practice problems** for each lecture
- For each project, make sure to fill out the **AI usage reflection** (cs.uw.edu/473/projects)

ASSIGNMENTS

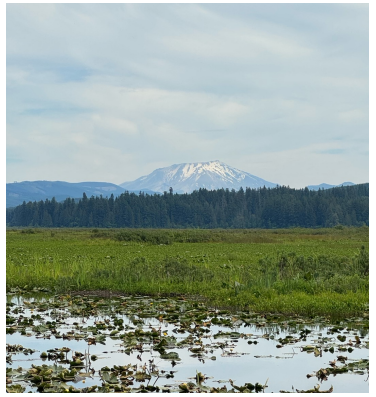
- **Homework 1** released today
  - **due** next Thursday (7.2)
- **Project 1** released today
  - **due** in two weeks (7.9)

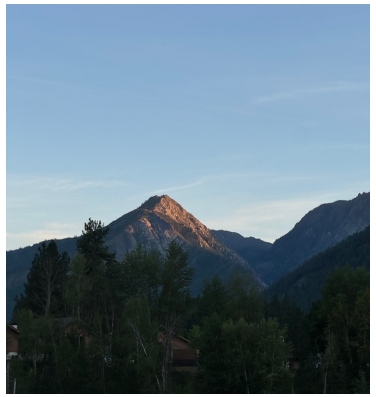
Search

FOUNDATIONS

James' Search Problem

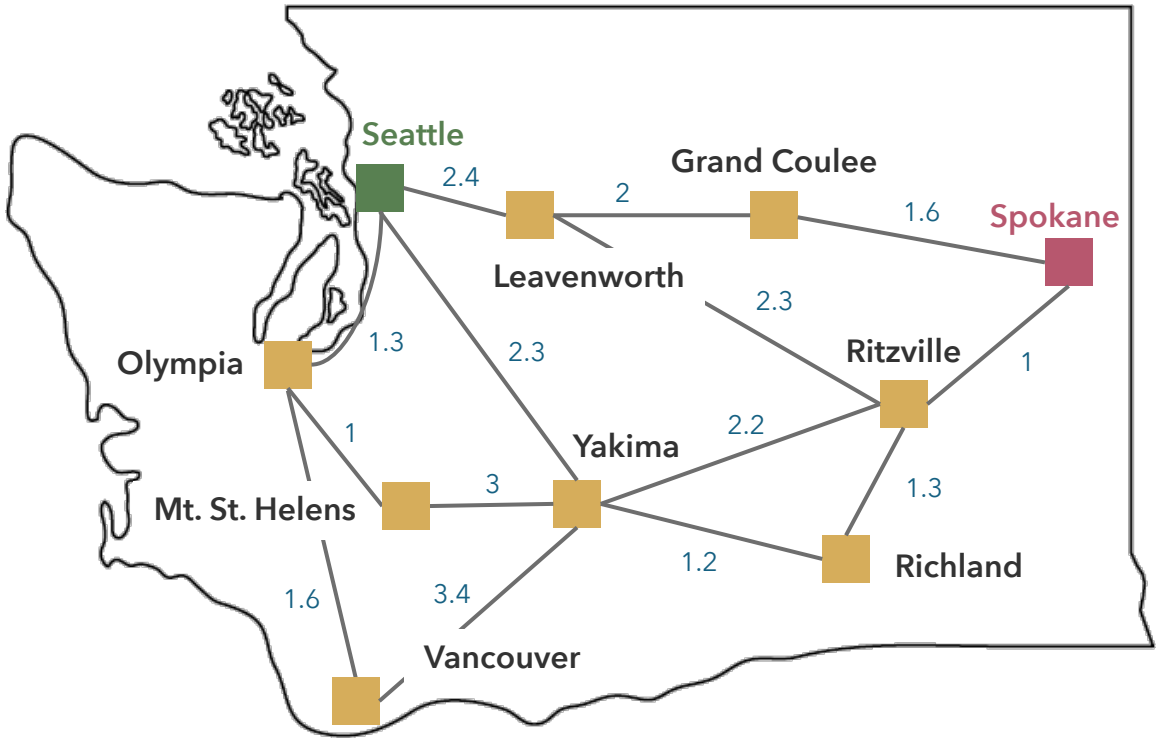
CONTEXT





Spokane to Seattle

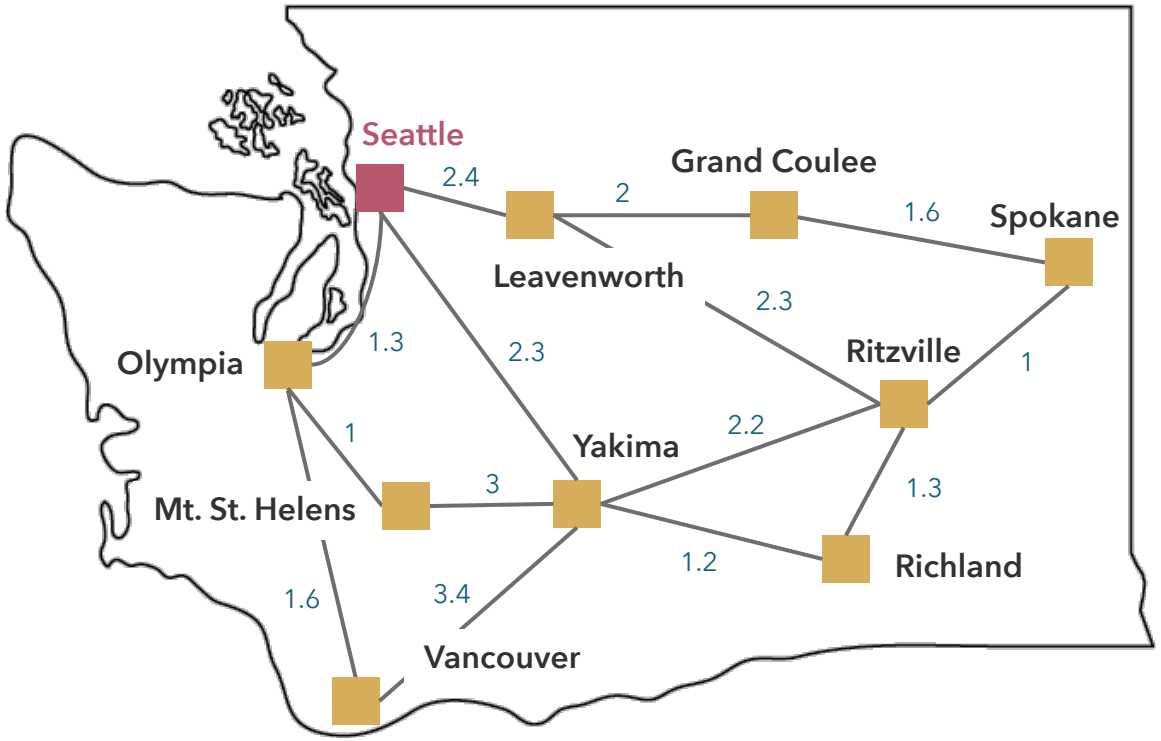
EXAMPLE



- state space
  - cities
- successor function
  - roads connect adjacent cities with cost = travel time
- start state
  - **Spokane**
- goal test
  - state == **Seattle**?

Touring Washington

EXAMPLE



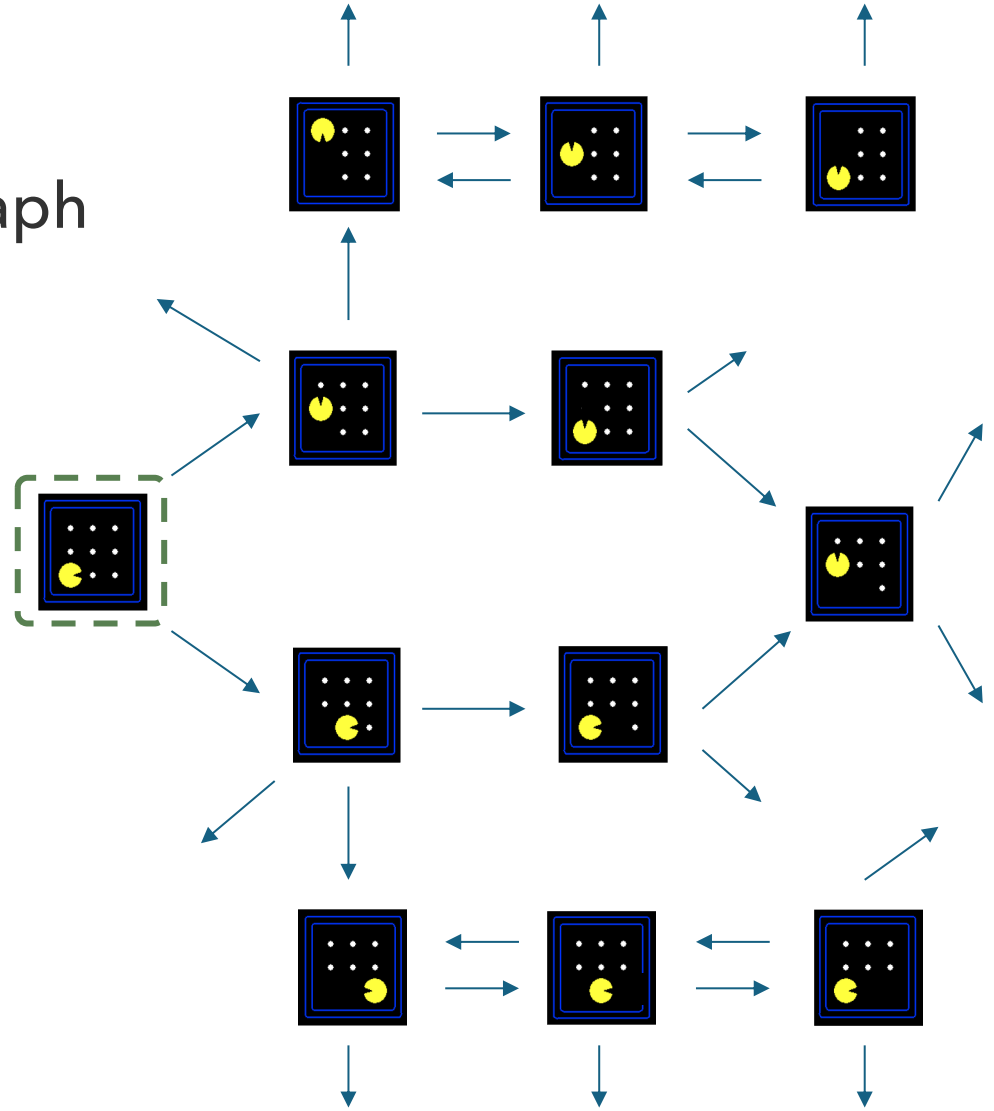
- state space
  - cities (`visited` boolean array)
- successor function
  - roads connect adjacent cities with cost = travel time
- start state
  - **Seattle**
- goal test
  - `visited` is all **`true`** and city == **Seattle**

State Space Graphs

FOUNDATIONS

Represent transitions between states as a graph

- **Nodes** are states
- **Arcs** represent successors (action results)
- **Goals** are a set of nodes
- Each state is represented once



Can we build the full graph?

Can't usually fit in memory

Search Trees

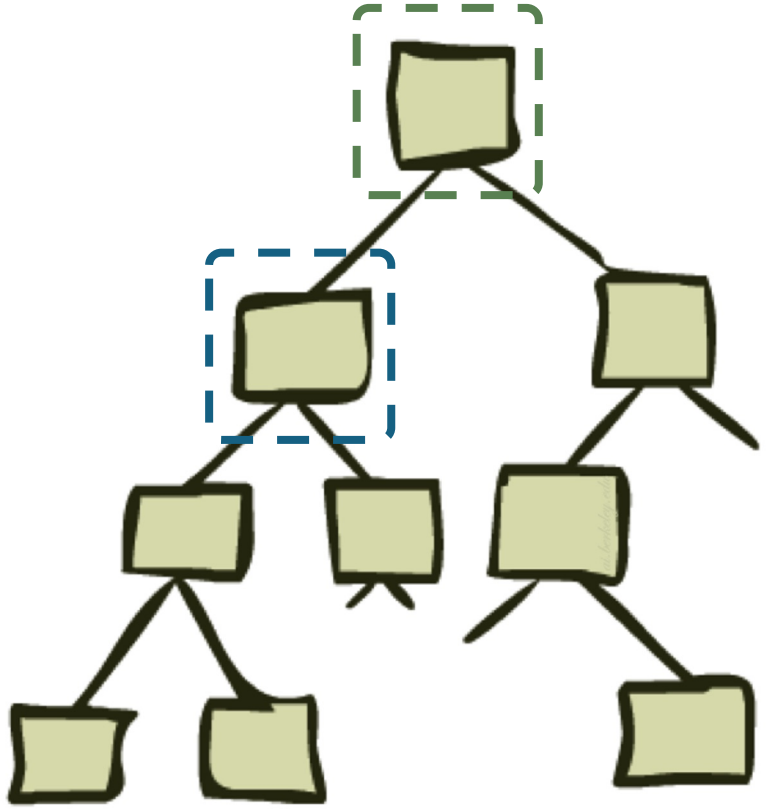
FOUNDATIONS

Represent sequences of actions on paths to root

- **Nodes** are states
- **Children** represent successors (action results)
- States *can* be repeated

Can we build the full tree?

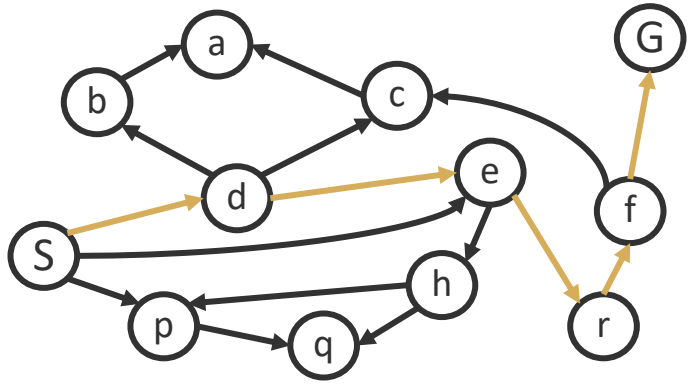
Not always possible…



Graphs vs. Trees

FOUNDATIONS

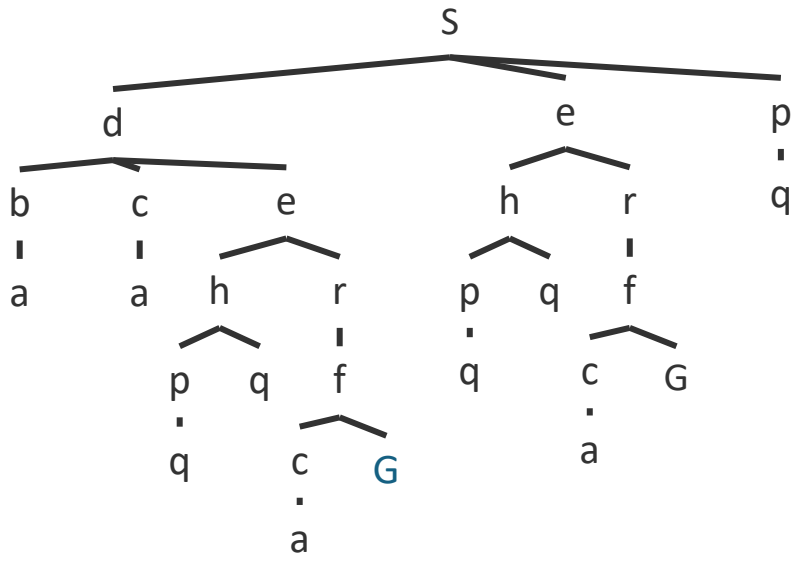
State Space Graph



Each **node** in a search tree represents an **entire path** from the root.

Trees get very large, so we construct on demand and as little as possible.

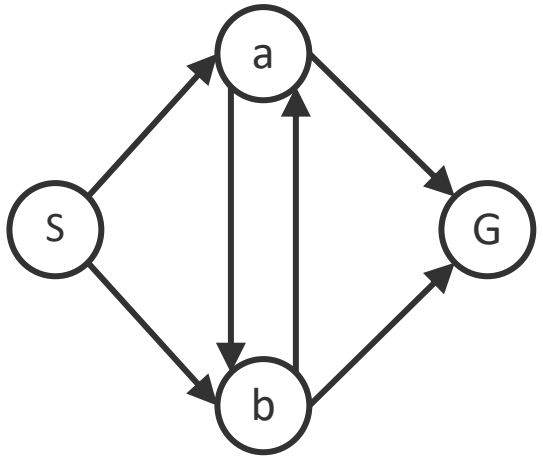
Search Tree

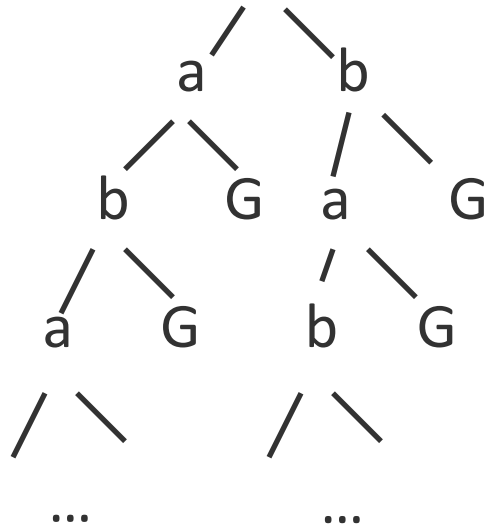


Graph vs. Tree

EXAMPLE

*The tree is **infinitely big** because the graph has a cycle.*





QUESTIONS

Questions?

live and on sli.do #cse473

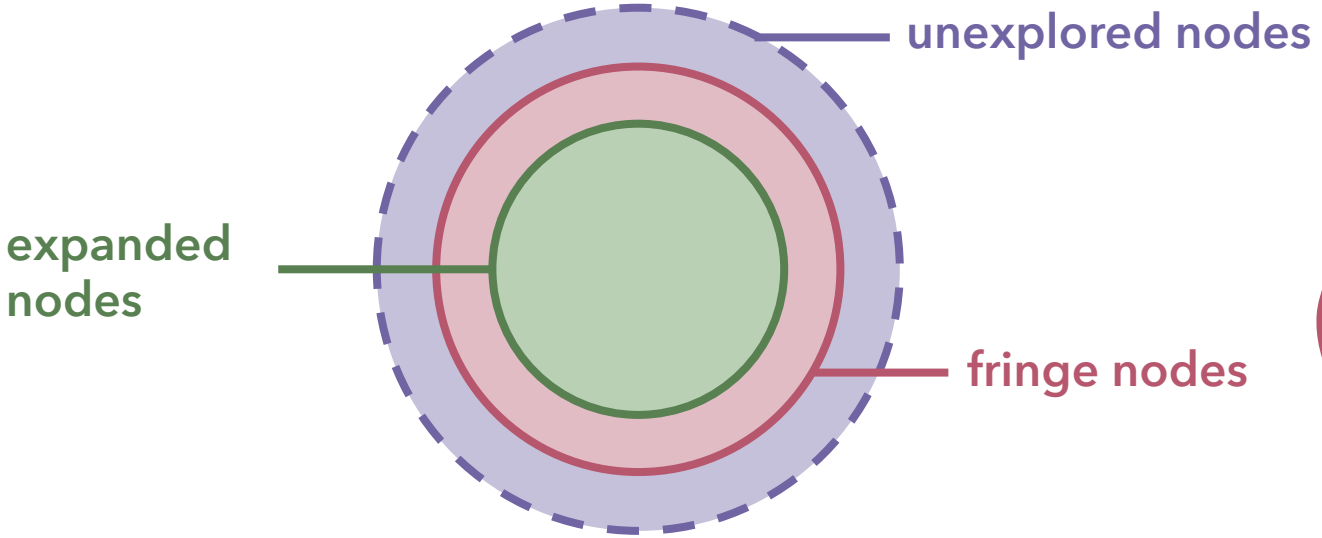
Tree Search

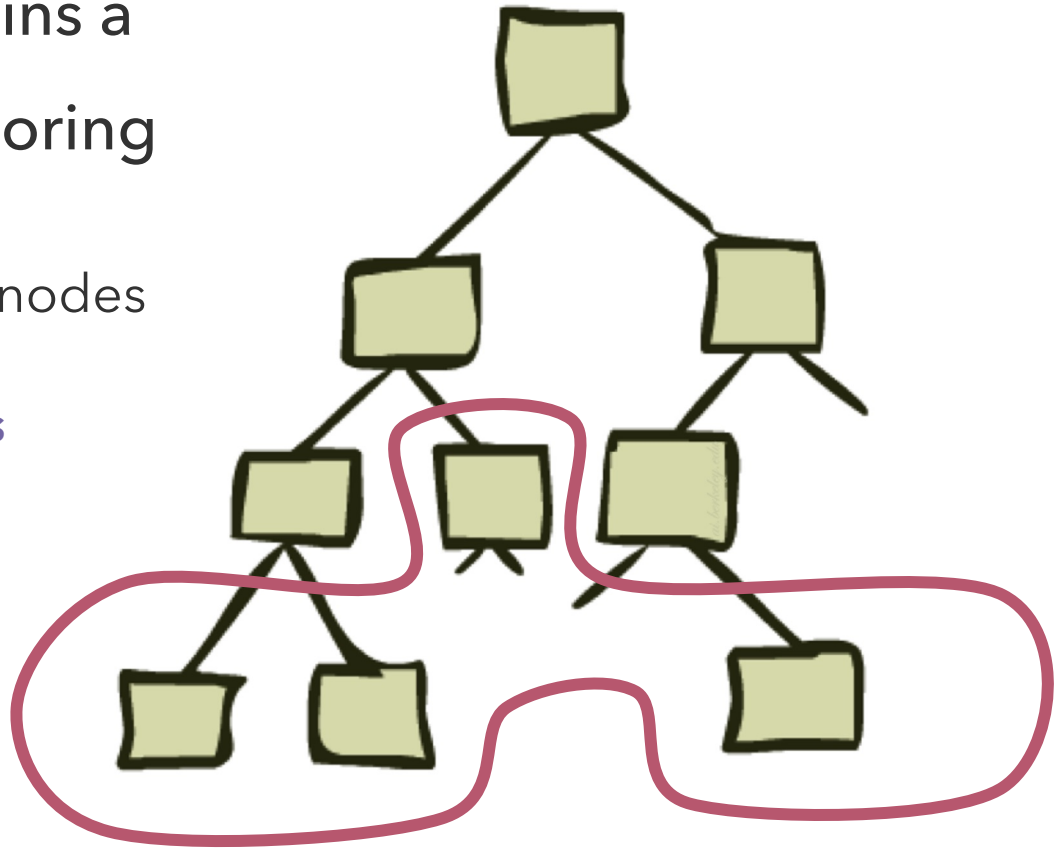
DEFINITION

Tree search is a search algorithm that maintains a

fringe (or frontier) of nodes to consider exploring

- Nodes on the fringe are ***expanded*** to find new nodes





General Tree Search

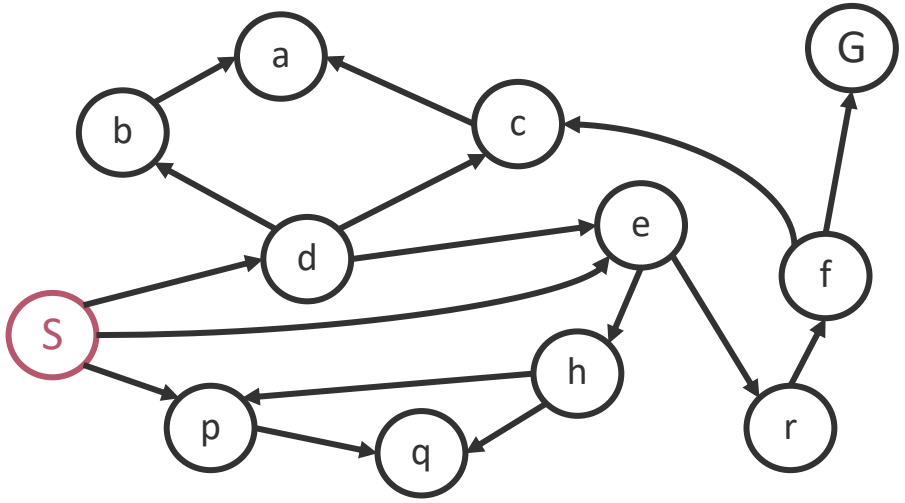
ALGORITHM

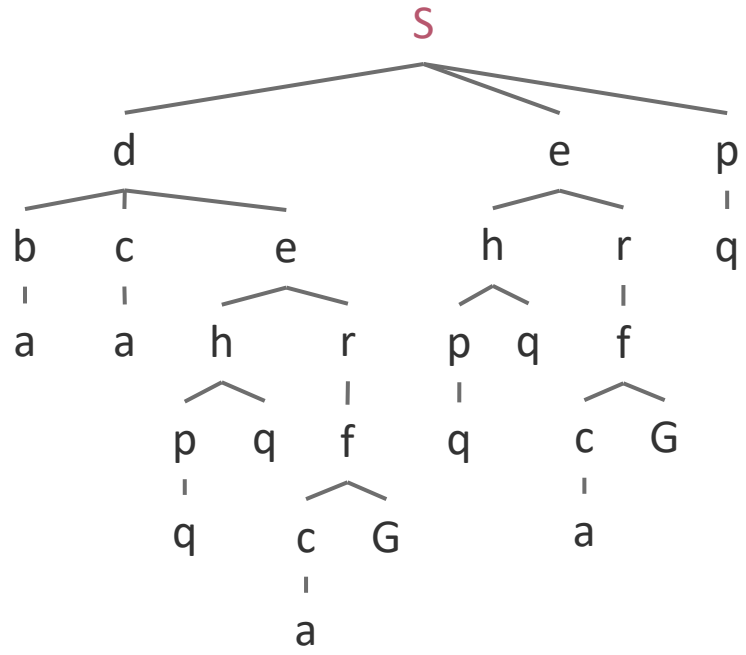
```
function TREE-SEARCH(problem, strategy) returns solution or failure
  initialize tree using initial state of problem
  loop do
        if no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if node contains goal state then return corresponding solution
        else expand node and add resulting nodes to search tree
  end
```

How to order fringe?

Random Exploration: Step 1

EXAMPLE





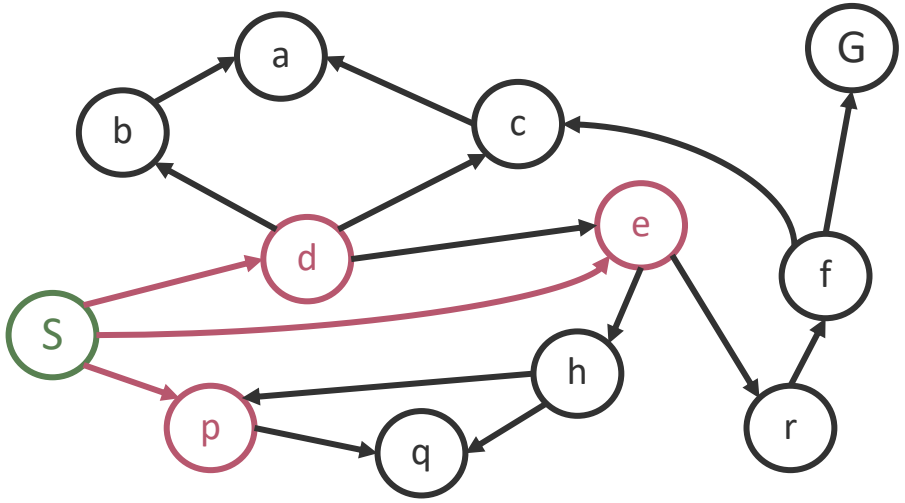
Expanded

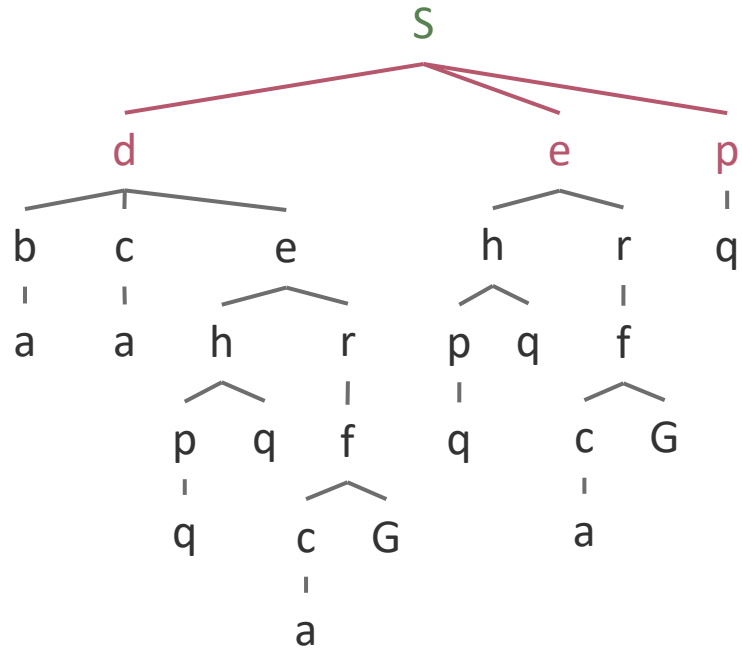
Fringe

S

Random Exploration: Step 2

EXAMPLE





Expanded

S

Expand

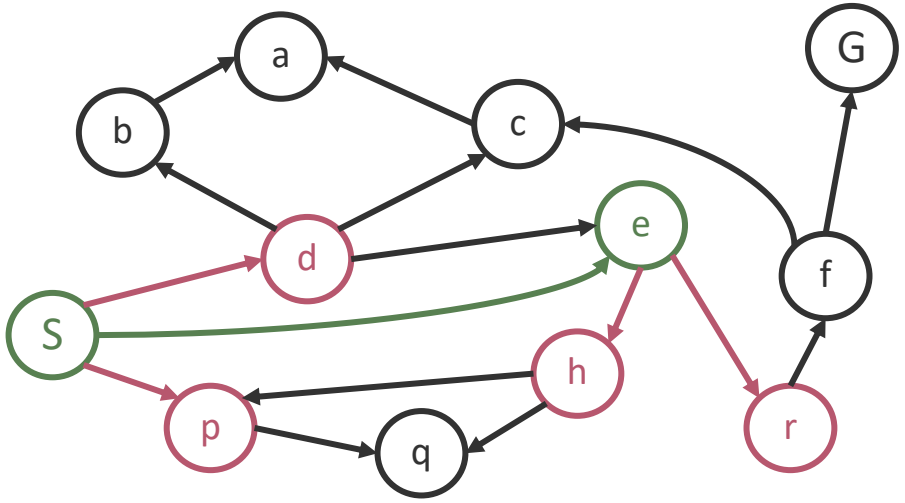
s

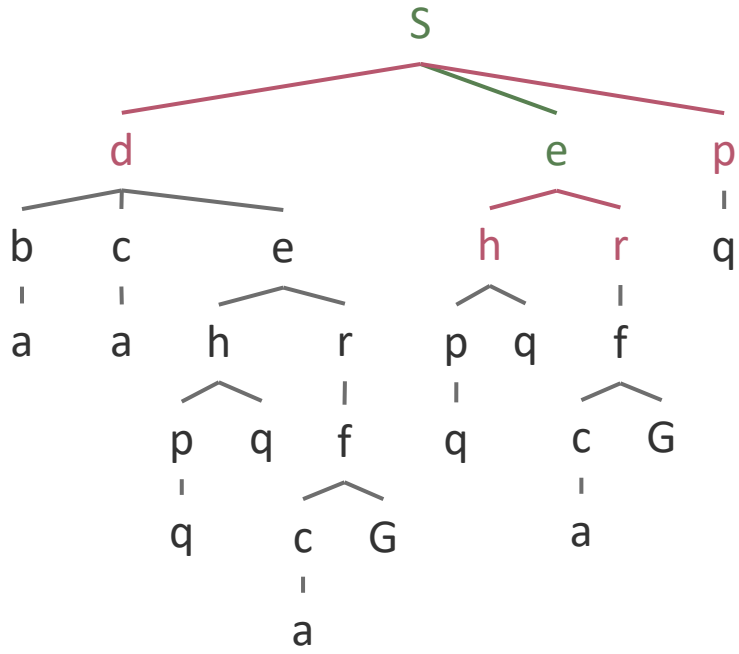
Fringe

d, e, p

Random Exploration: Step 3

EXAMPLE





Expanded

S, e

Expand

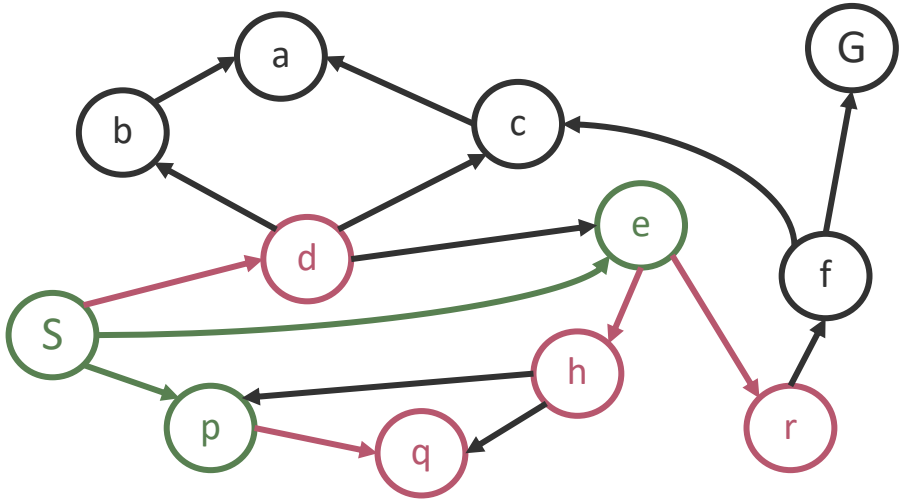
e

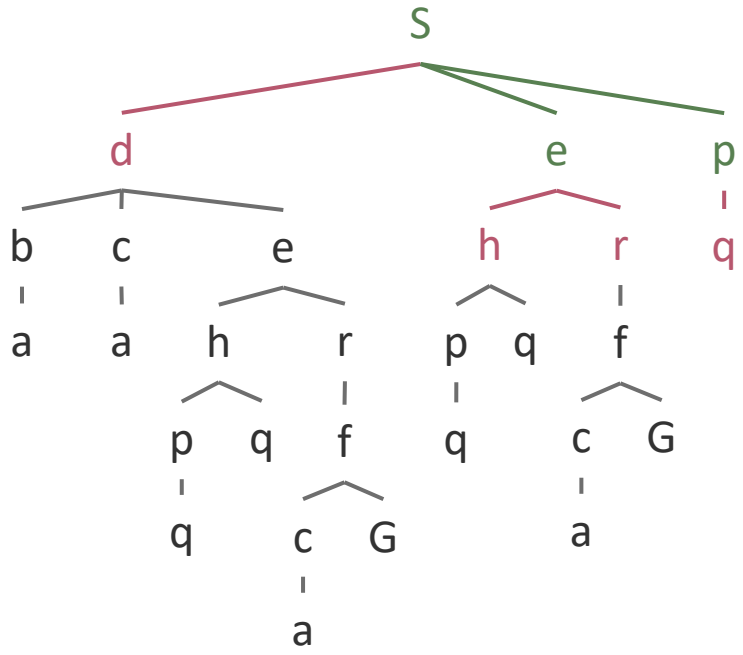
Fringe

d, p, h, r

Random Exploration: Step 4

EXAMPLE





Expanded

S, e, p

Expand

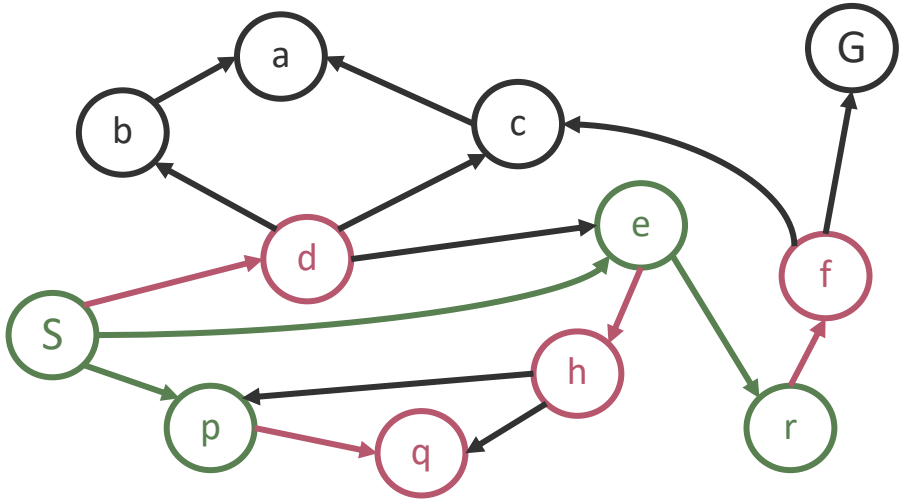
p

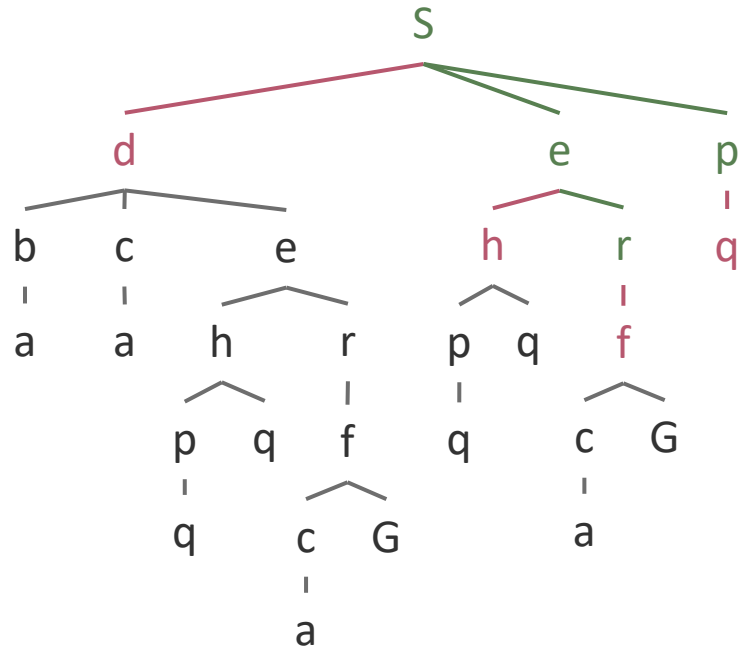
Fringe

d, h, r, q

Random Exploration: Step 5

EXAMPLE





Expanded

S, e, p, r

Expand

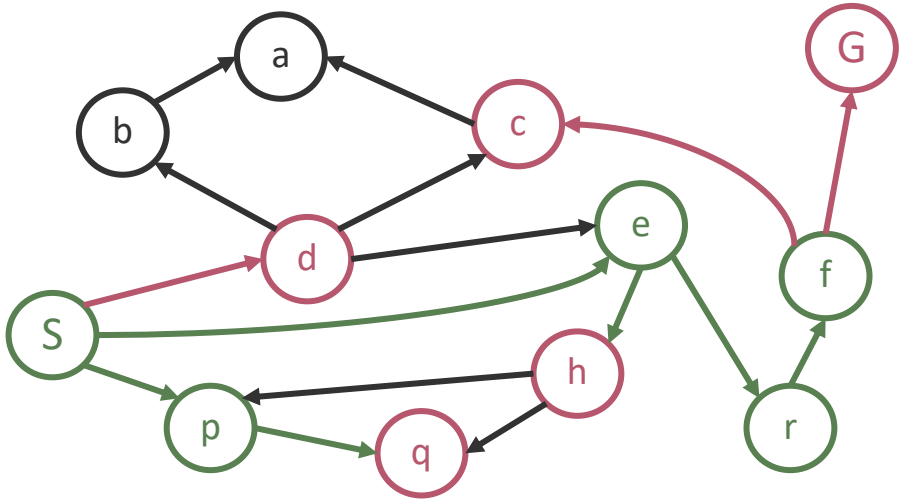
r

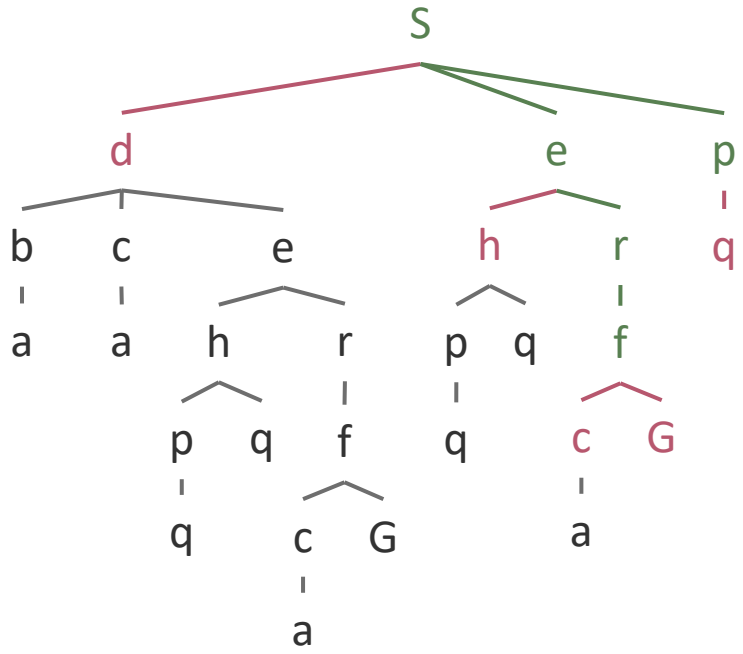
Fringe

d, h, q, f

Random Exploration: Step 6

EXAMPLE





Expanded

S, e, p, r, f

Expand

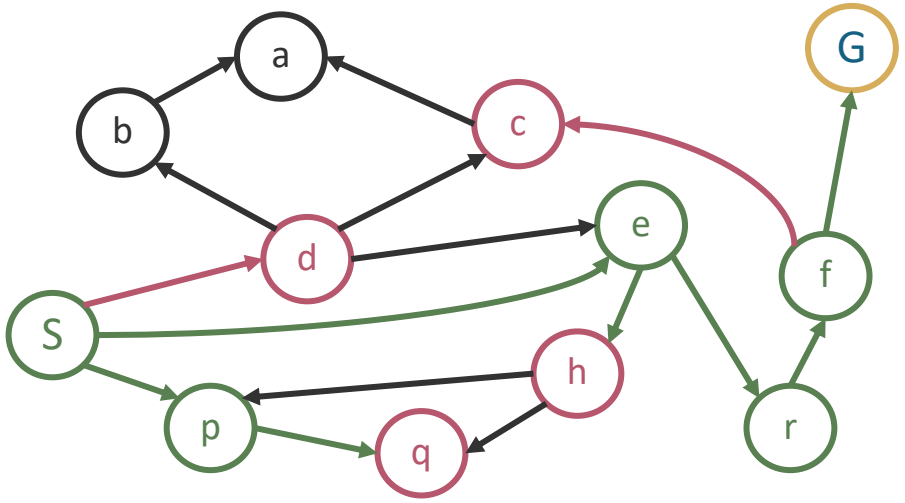
f

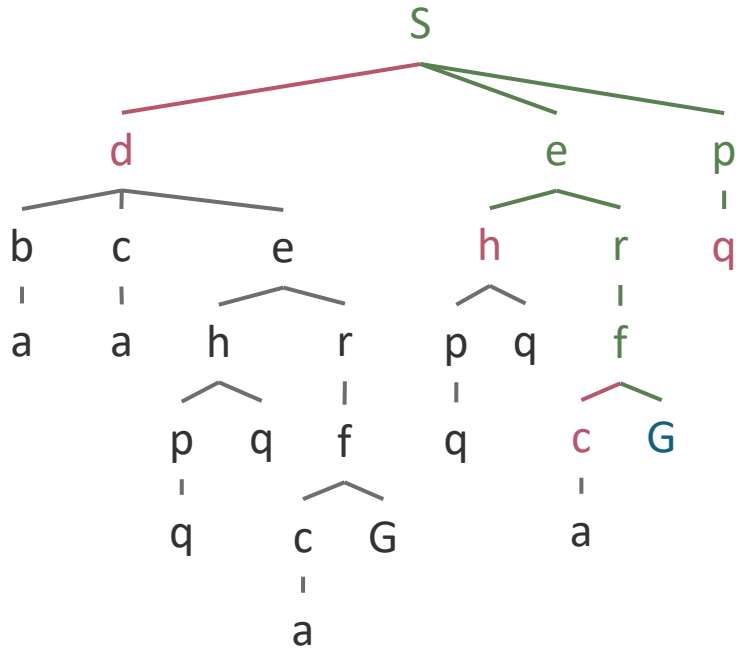
Fringe

d, h, q, c, G

Random Exploration: Step 7

EXAMPLE





Expanded

S, e, p, r, f

"Expand"
G

Fringe

d, h, q, c

QUESTIONS

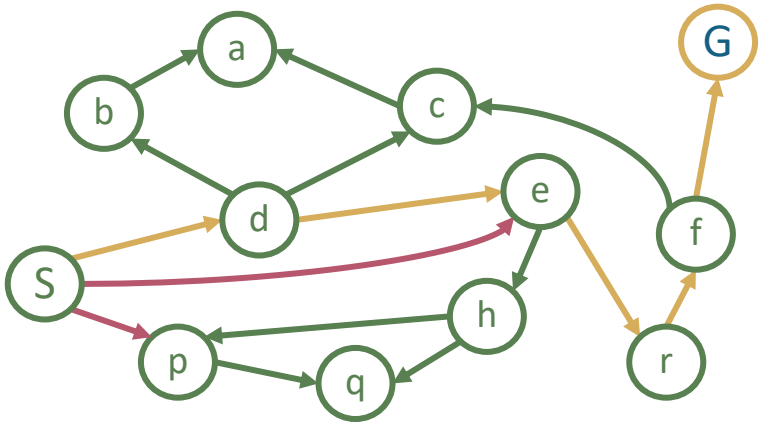
Questions?

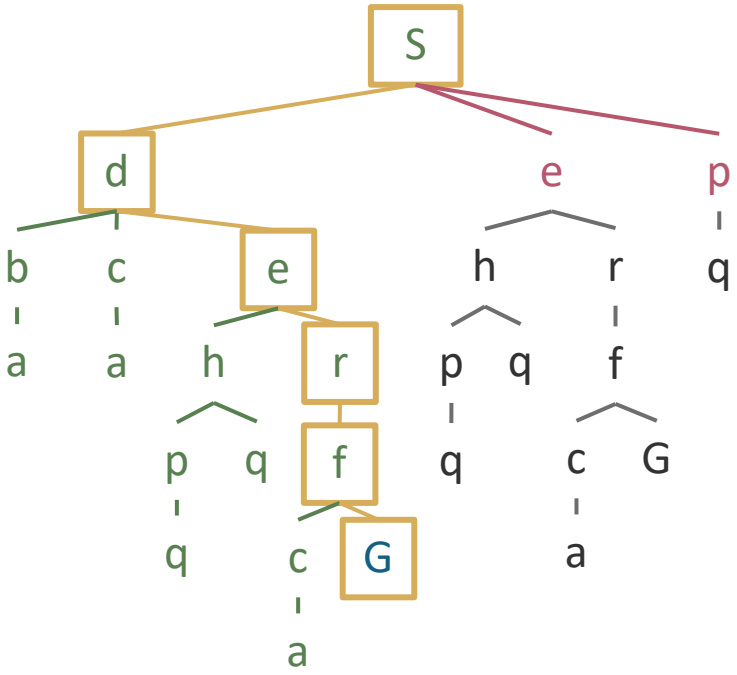
live and on sli.do #cse473

Depth-First Search

FOUNDATIONS

- Strategy
  - Expand **deepest** (most recently added) node first
- Implementation
  - Fringe is a **LIFO stack**
  - Usually break ties alphabetically



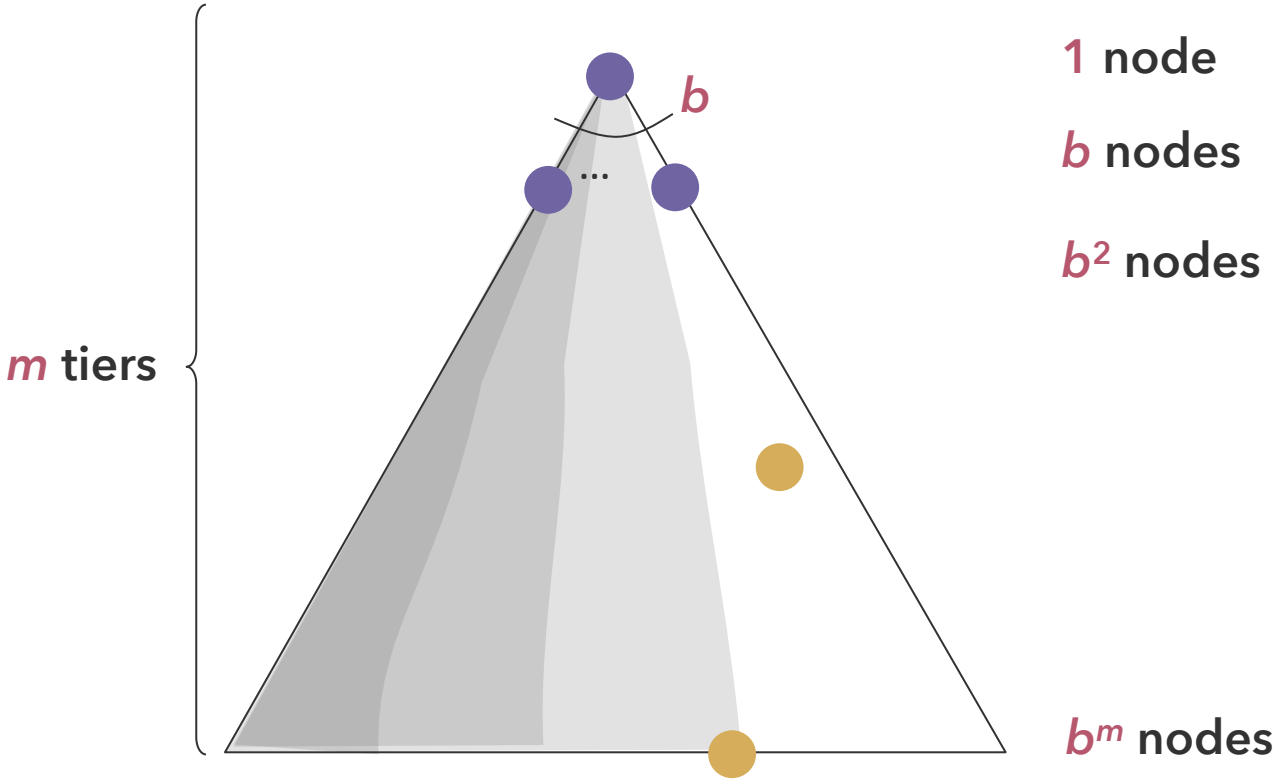


"oooooo, shiny!"

Properties of DFS

DEFINITION

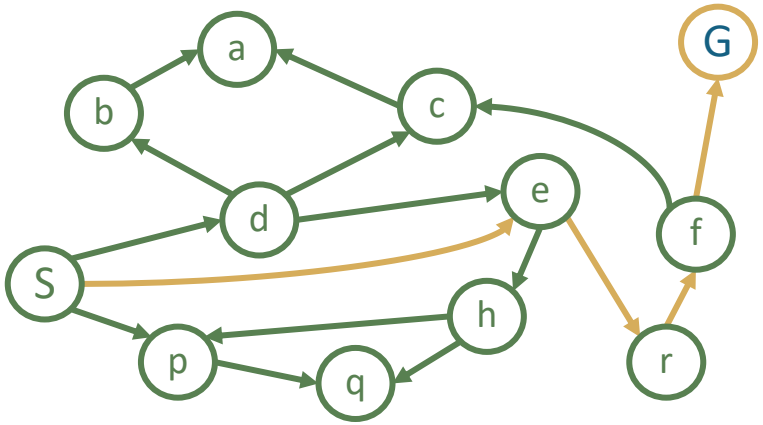
- Complete?
  - *m* could be infinite, so only if no cycles
- Optimal?
  - No. DFS does not care about cost!
- Time Complexity
  - $O(b^m)$ if *m* is finite
- Space Complexity
  - $O(bm)$

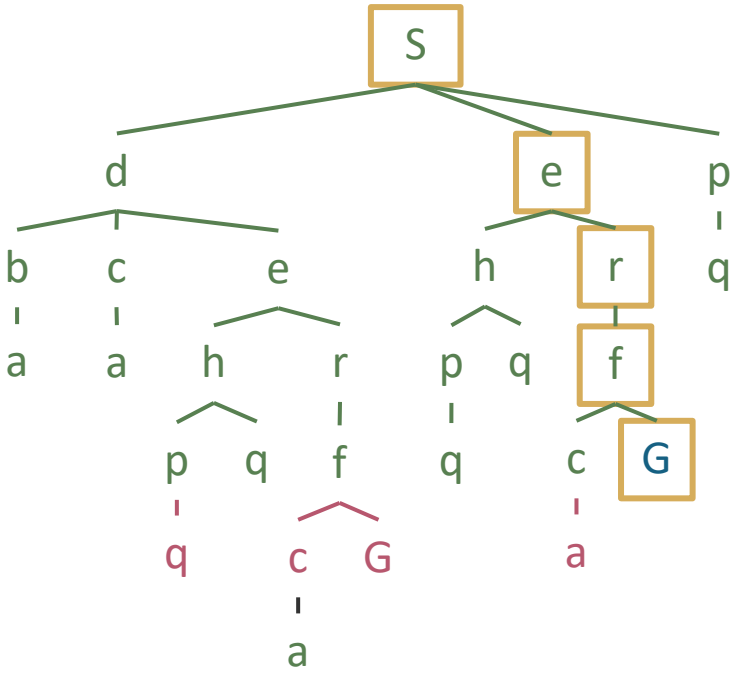


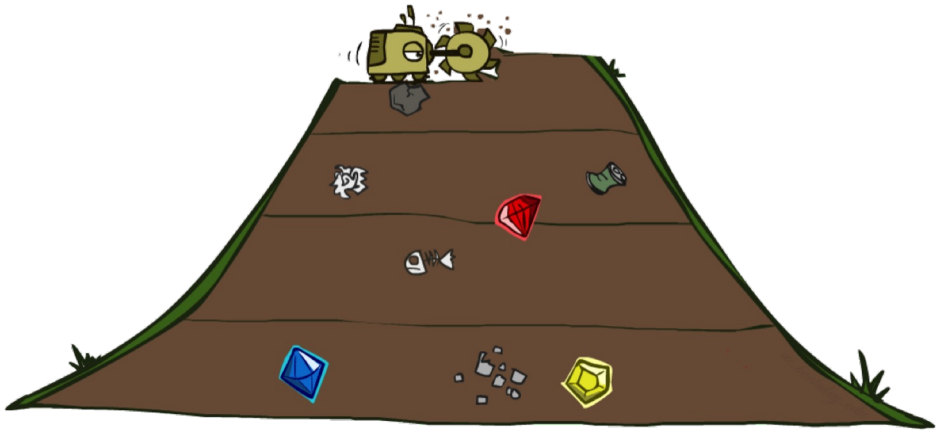
Breadth-First Search

FOUNDATIONS

- Strategy
  - Expand **shallowest** (oldest) node first
- Implementation
  - Fringe is a **FIFO queue**
  - Usually break ties alphabetically







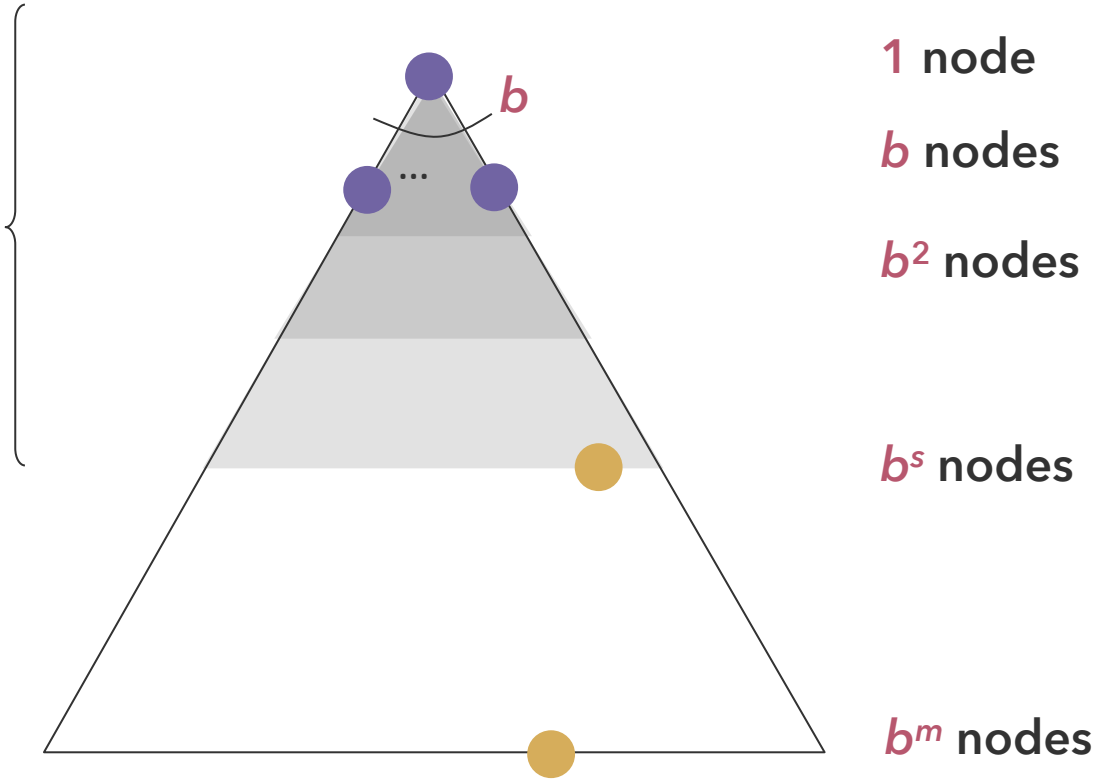
"wait your turn!"

Properties of BFS

DEFINITION

s = depth of shallowest solution

- Complete?
  - Yes (s must be finite for a solution to exist)
- Optimal?
  - Only if costs are all 1
- Time Complexity
  - $O(b^s)$
- Space Complexity
  - $O(b^s)$ (roughly all nodes in tier s)



DFS vs. BFS?

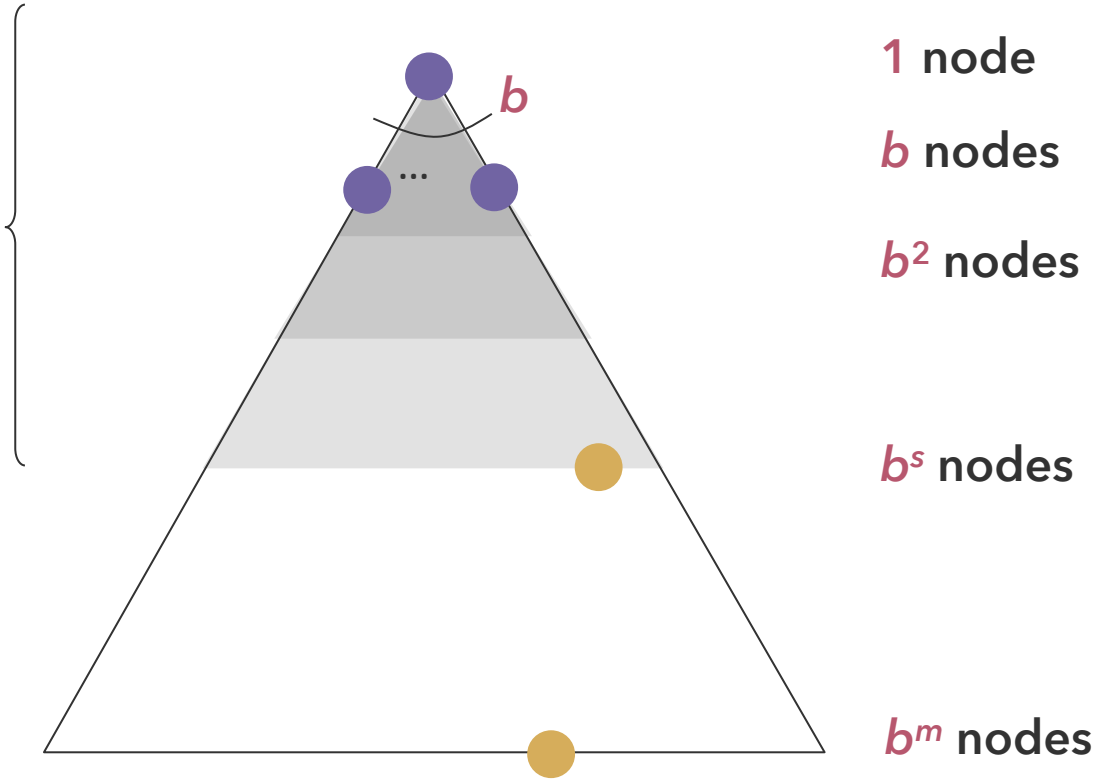
Iterative Deepening

FOUNDATIONS

s = depth of shallowest solution

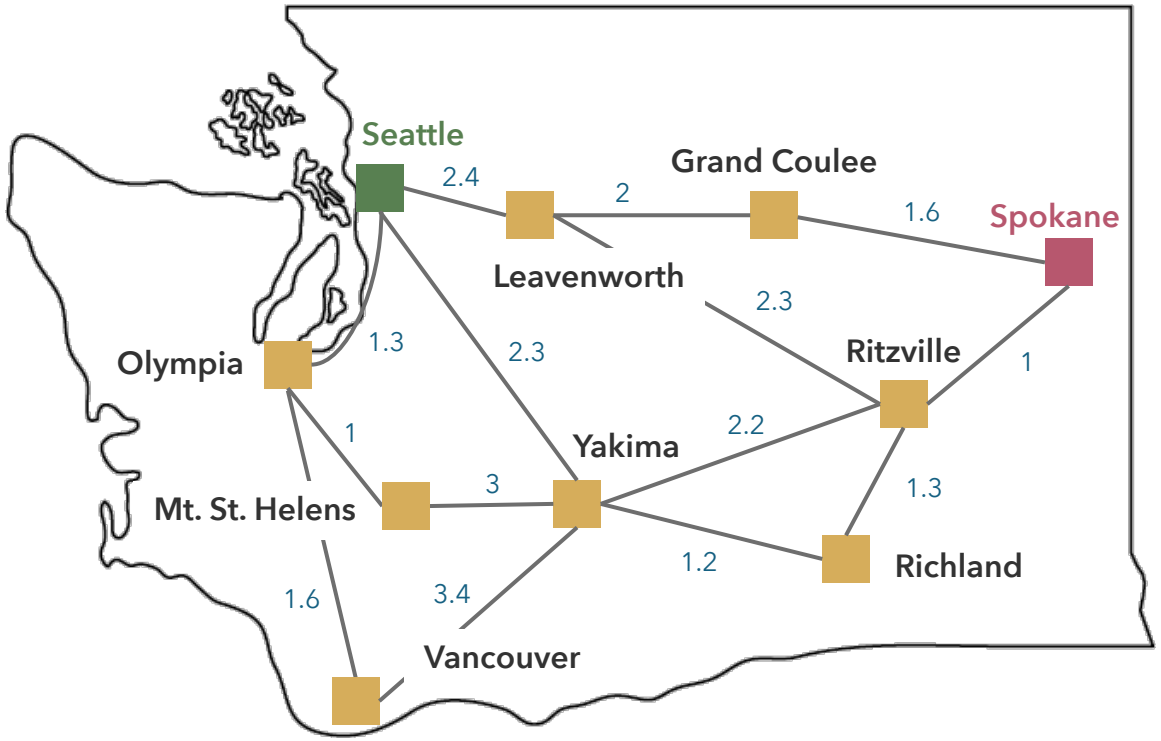
- Idea
  - Combine space advantage of DFS with BFS time advantage
- Implementation
  - Run DFS with depth 1
  - Run DFS with depth 2
  - Run DFS with depth 3

  …



Actions Have Costs…

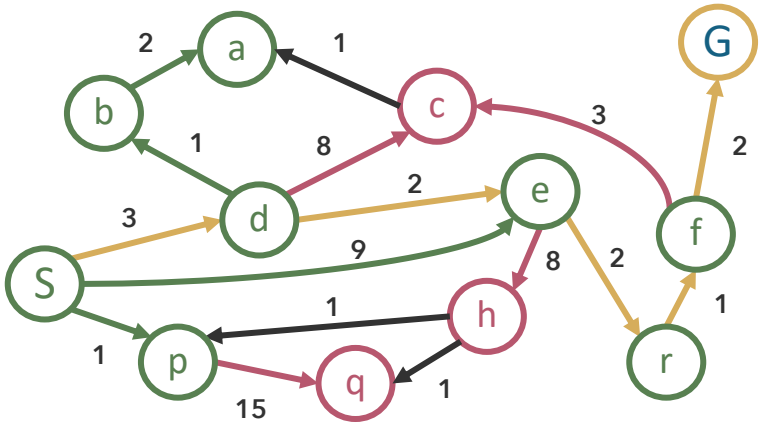
EXAMPLE



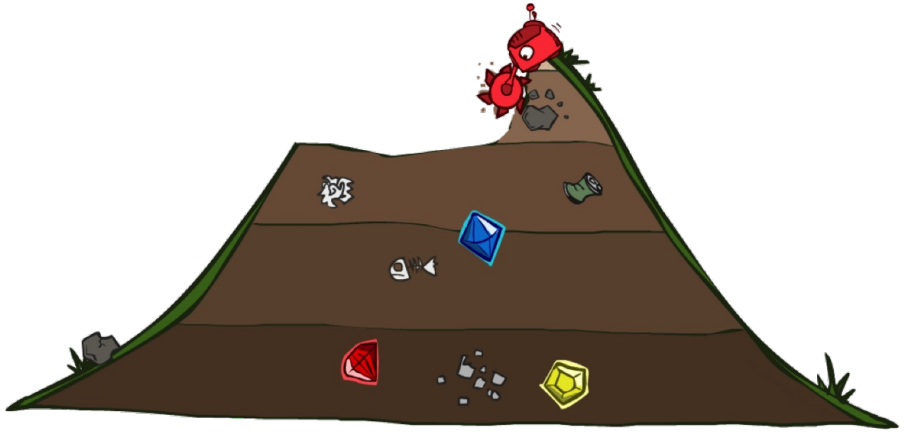
How to find the lowest cost path?

Uniform Cost Search

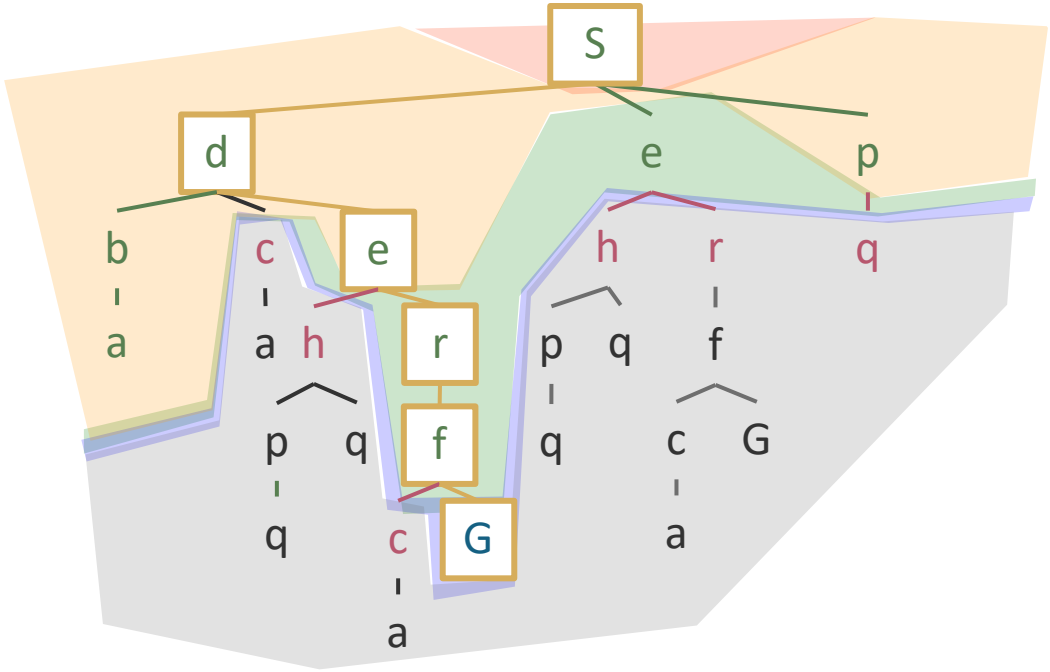
FOUNDATIONS



- Strategy
  - Expand **cheapest** (least cost) node first
- Implementation
  - Fringe is a **priority queue**
  - Usually break ties alphabetically



"how much does it cost?"

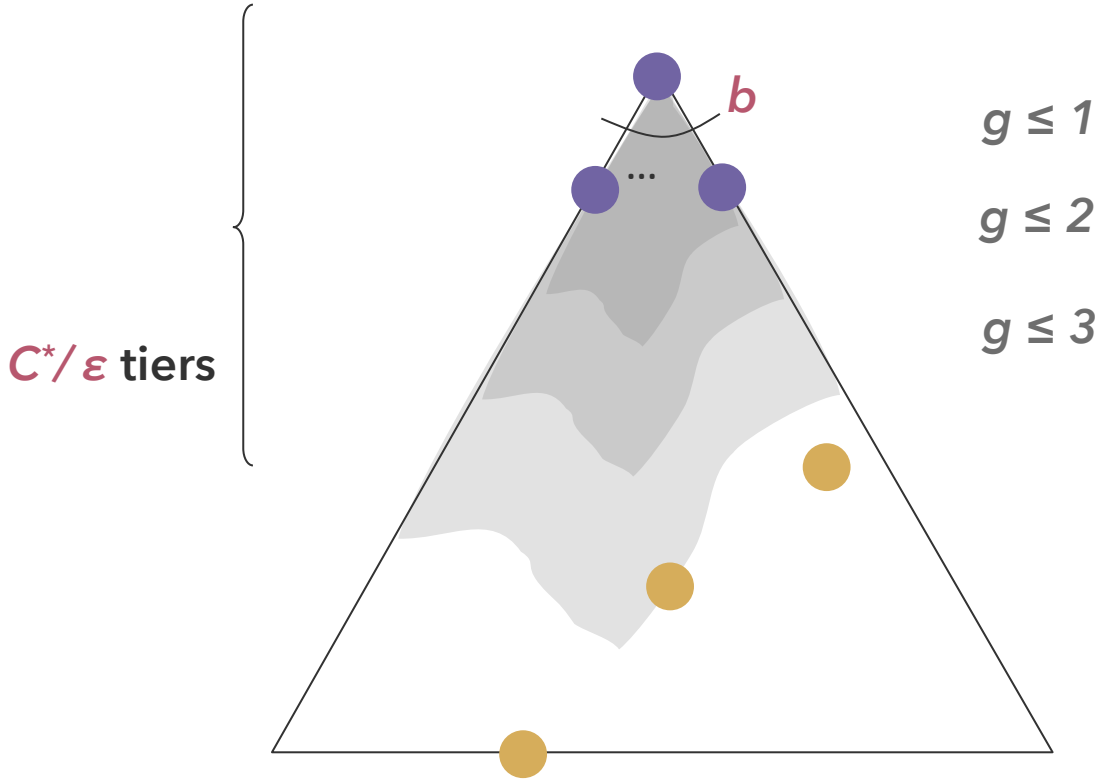


Properties of UCS

DEFINITION

$g(n)$ = cost from root to node n
C^* = solution cost
ε = minimum arc cost
effective depth = C^*/ε

- Complete?
  - Yes (assuming C^* is finite and $\varepsilon > 0$)
- Optimal?
  - Yes!
- Time Complexity
  - $O(b^{C^*/\varepsilon})$
- Space Complexity
  - $O(b^{C^*/\varepsilon})$ (roughly last tier)



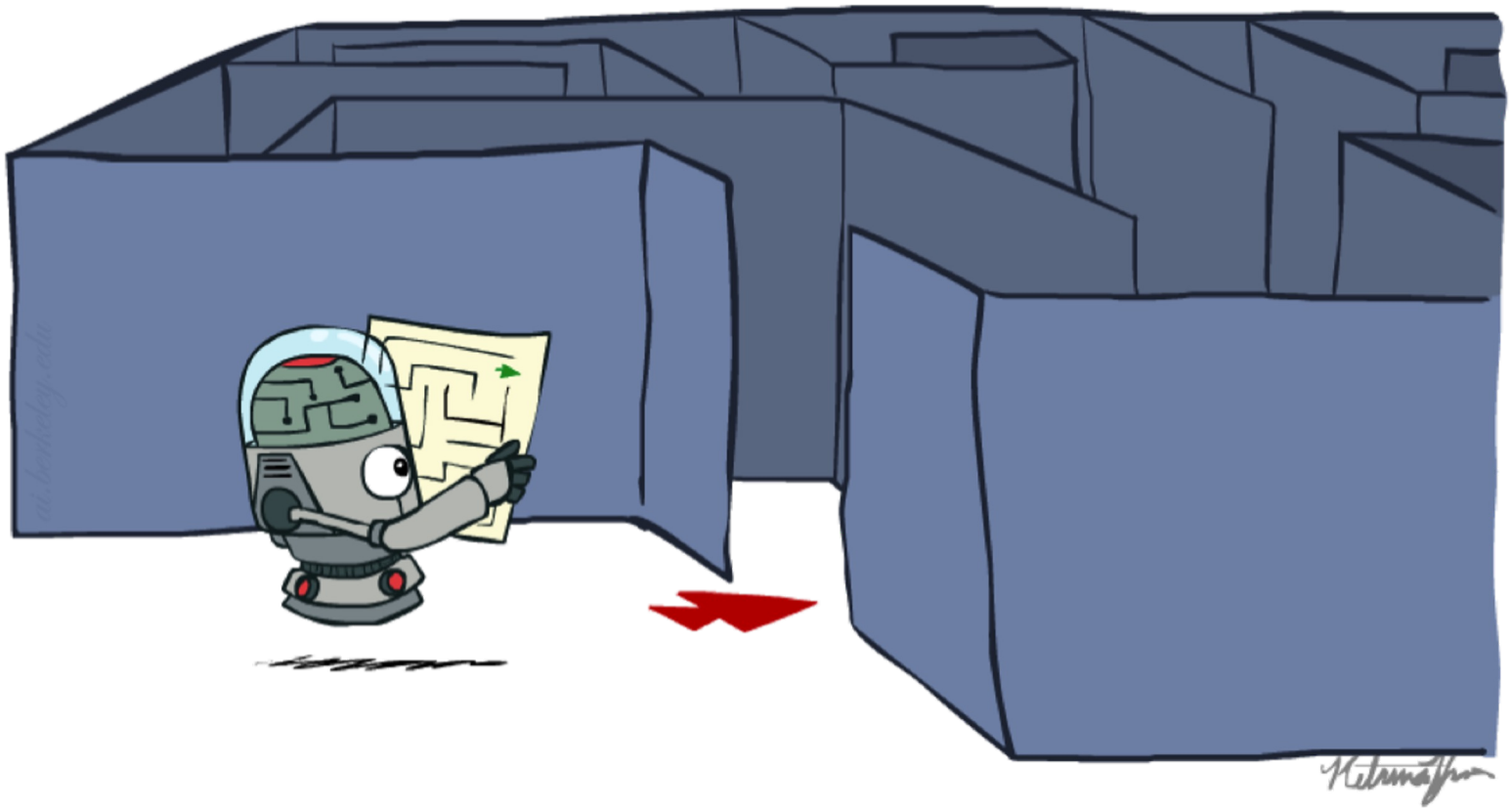
QUESTIONS

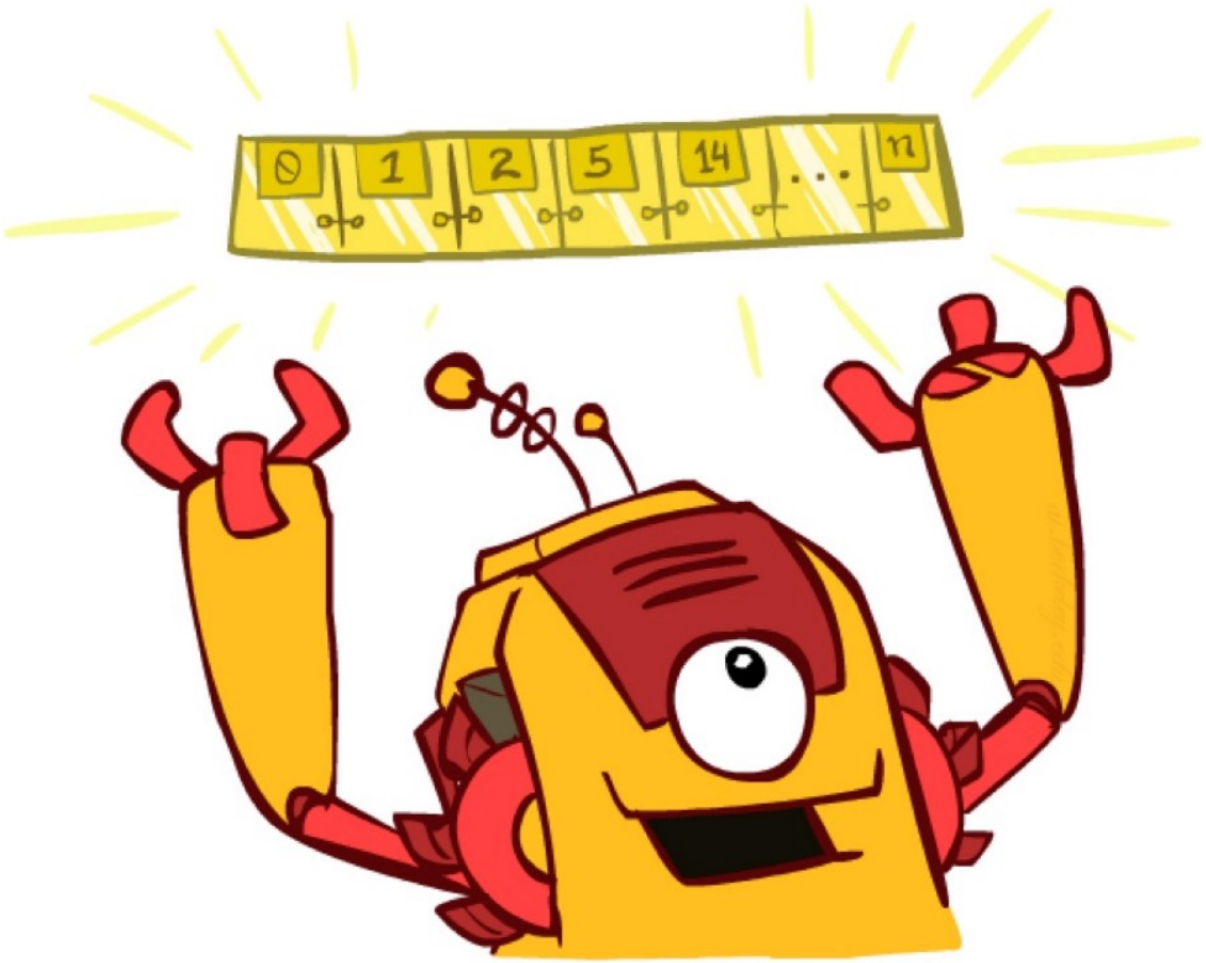
Questions?

live and on sli.do #cse473

Pacman Wayfinding

EXAMPLE





The One (Priority) Queue

General Graph Search

ALGORITHM

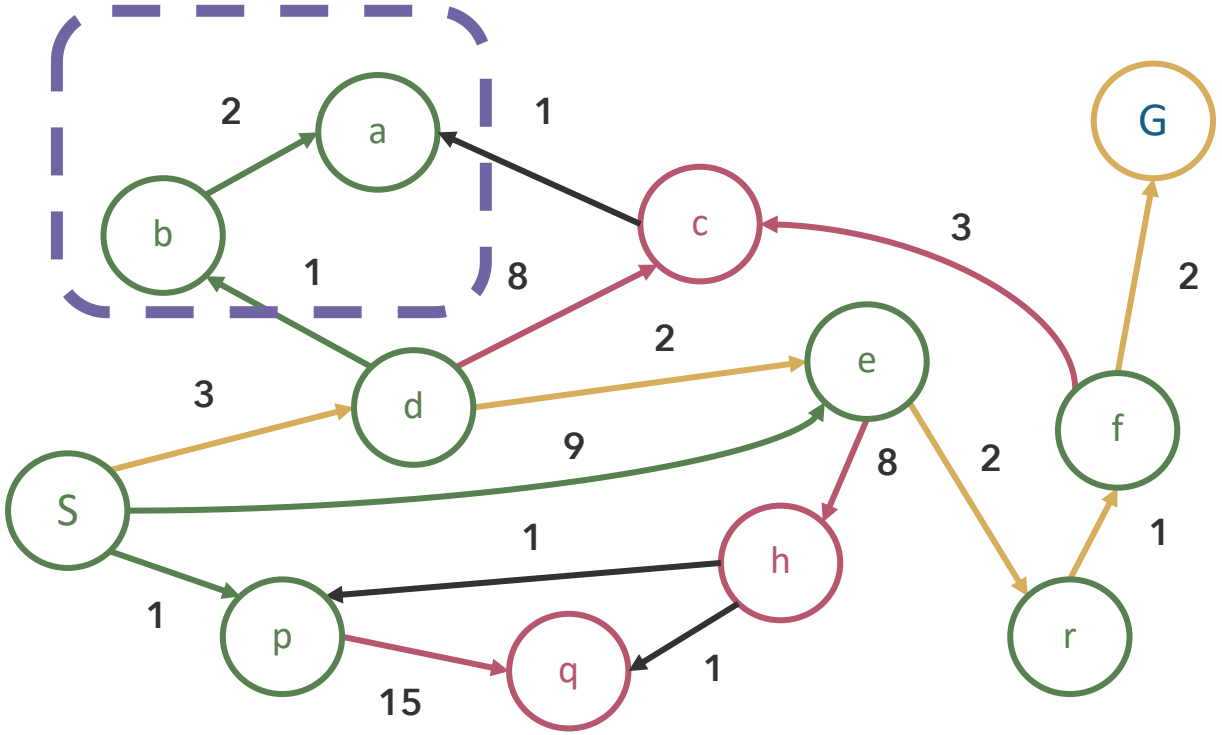
```
function GRAPH-SEARCH(problem, strategy) returns solution or failure
  closed ← empty set
  initialize fringe using initial state of problem
  loop do
        if fringe empty then return failure
        node ← remove node from fringe
        if node is goal state then return corresponding solution
        if node not in closed then
                expand node and add resulting nodes to fringe
                add node to closed
  end
```

What's Next?

CONTEXT

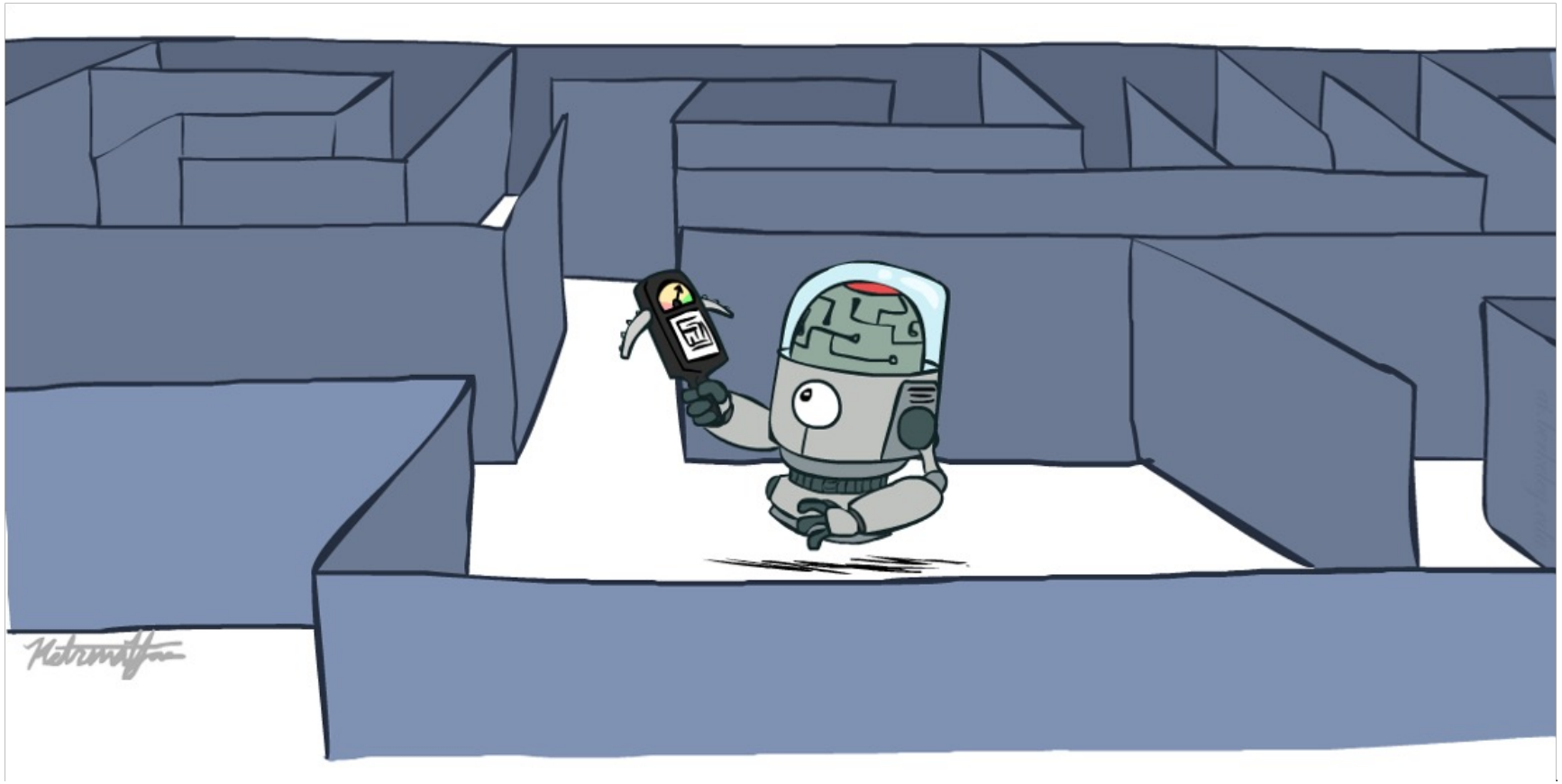
dead end

can we avoid it?



Can We *Inform* Our Search?

CONTEXT



that's it for today!

SUMMARY

- *Which fringe nodes to expand?*
- DFS (stack) vs. BFS (queue) vs. UCS (priority queue)

UPCOMING

- Search with Heuristics
- Multi-Agent Search

REMINDERS

- Complete **Practice Problem 3**
- Do **Homework 1** (due 7.2)
- Do **Project 1** (due 7.9)