Lecture 2: Search

CS 486/686: Introduction to Artificial Intelligence

Plan for Today

- Uniformed Search Methods
- Thinking about the importance of abstractions

Introduction

Search was one of the first topics studied in AI

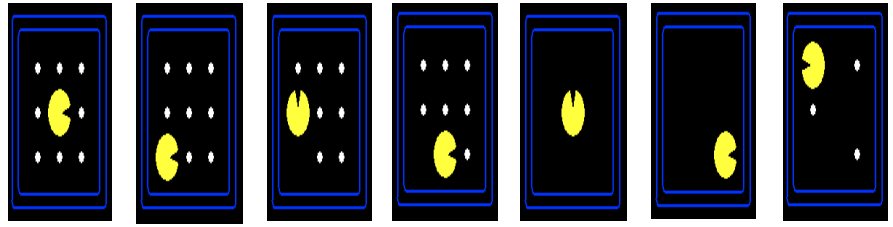
- Newell and Simon (1961) *General Problem Solver*

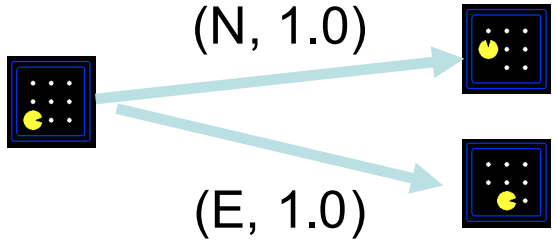
Central component to many AI systems

- Automated reasoning, theorem proving, robot navigation, scheduling, game playing, machine learning...

Search Problems

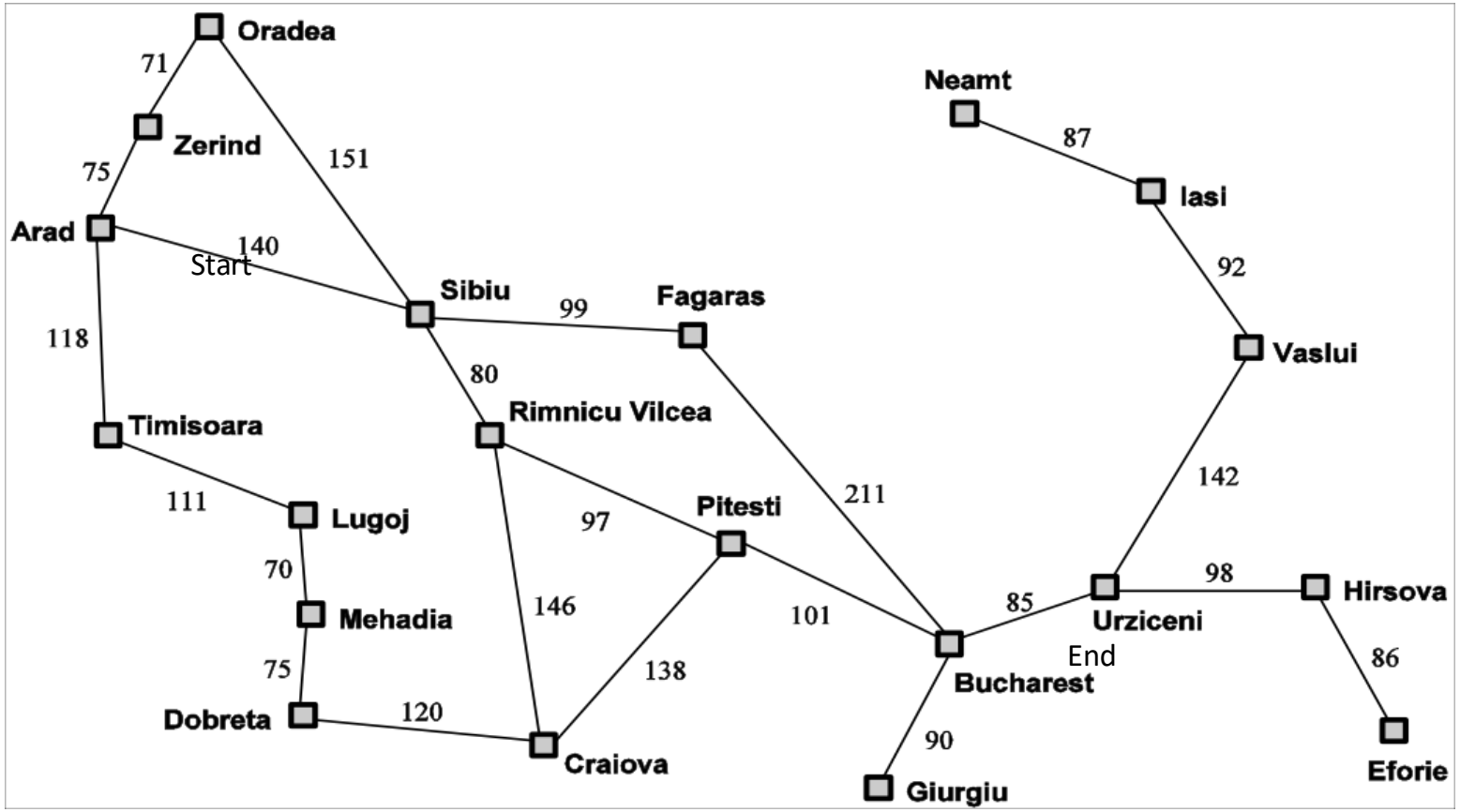
- A **search problem** consists of
  - a state space
  - a successor function (actions, cost)





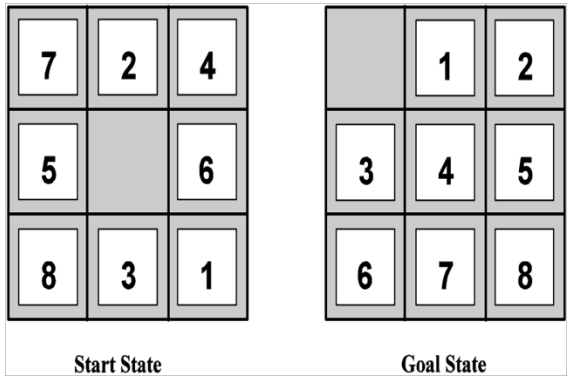
  - a start state and a goal test
- A **solution** is a sequence of actions (plan) from the start state to a goal state

Example: Traveling in Romania



- States:
- Initial State:
- Successor Function:
- Goal test:
- Solution:

Examples of Search Problems



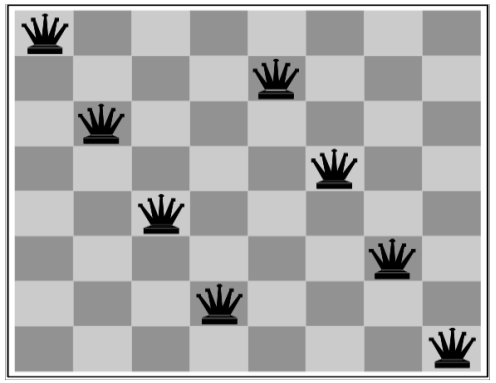
States:

Initial State:

Successor Function:

Goal test:

Solution:



States:

Initial State:

Successor Function:

Goal test:

Solution:

Examples of Search Problems

Subject	Catalog #	Units	Title
CS	480	0.5	Intro Machine Learning

Notes: For information on CS course enrollment during the Add/Drop period, see https://cs.uwaterloo.ca/computer-science/current-undergraduate-students/majors/cs-course-enrollment

Class	Comp	Sec	Camp Loc	Assoc	Class	Rel 1	Rel 2	Enrl Cap	Enrl Tot	Wait Cap	Wait Tot	Time	Days/Date	Bldg	Room	Instructor
6328	LEC	001	UW U		1			63	63	0	0	04:00-05:20	MW			
Held With: CS 680 LEC 001																
6448	LEC	002	UW U		2			63	63	0	0	02:30-03:50	MW			
Held With: CS 680 LEC 002																

Subject	Catalog #	Units	Title
CS	484	0.5	Computational Vision

Notes: For information on CS course enrollment during the Add/Drop period, see https://cs.uwaterloo.ca/computer-science/current-undergraduate-students/majors/cs-course-enrollment

Class	Comp	Sec	Camp Loc	Assoc	Class	Rel 1	Rel 2	Enrl Cap	Enrl Tot	Wait Cap	Wait Tot	Time	Days/Date	Bldg	Room	Instructor
7896	LEC	001	UW U		1			68	68	0	0	01:00-02:20	MW			
Held With: CS 684 LEC 001																

Subject	Catalog #	Units	Title
CS	485	0.5	Machine Learning Foundations

Notes: For information on CS course enrollment during the Add/Drop period, see https://cs.uwaterloo.ca/computer-science/current-undergraduate-students/majors/cs-course-enrollment

Class	Comp	Sec	Camp Loc	Assoc	Class	Rel 1	Rel 2	Enrl Cap	Enrl Tot	Wait Cap	Wait Tot	Time	Days/Date	Bldg	Room	Instructor
6427	LEC	001	UW U		1			68	60	0	0	04:00-05:20	TTh			
Held With: CS 685 LEC 001																

Subject	Catalog #	Units	Title
CS	486	0.5	Intro Artificial Intelligence

Notes: For information on CS course enrollment during the Add/Drop period, see https://cs.uwaterloo.ca/computer-science/current-undergraduate-students/majors/cs-course-enrollment

Class	Comp	Sec	Camp Loc	Assoc	Class	Rel 1	Rel 2	Enrl Cap	Enrl Tot	Wait Cap	Wait Tot	Time	Days/Date	Bldg	Room	Instructor
6009	LEC	001	UW U		1	101		63	60	0	0	08:30-09:50	MW			
Held With: CS 686 LEC 001																
6264	LEC	002	UW U		2	101		63	63	0	0	10:00-11:20	MW			
Held With: CS 686 LEC 002																
6601	TST	101	UW U		9999			140	123	0	0	07:00-08:50 10/29-10/29	Th			
Held With: CS 686 TST 101																

Our Definition Excludes...

Chance

Adversaries

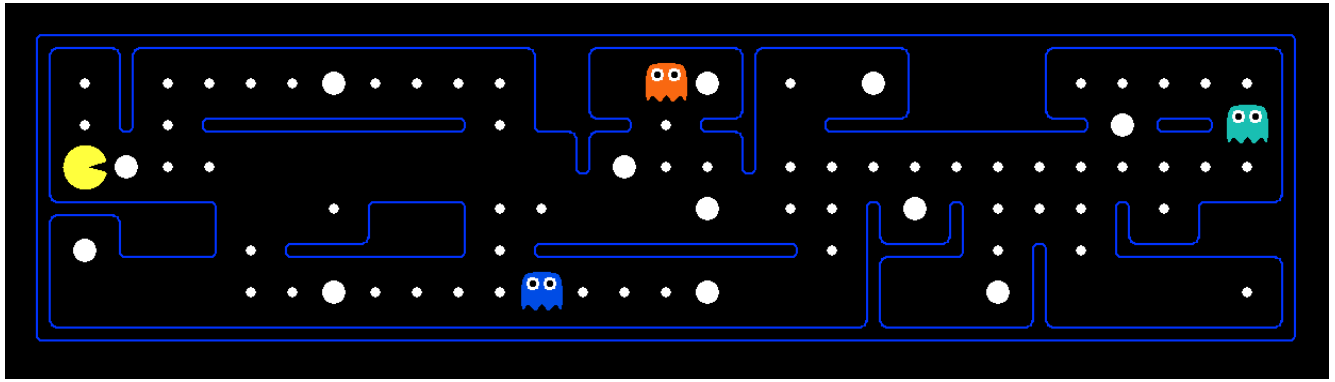
Continuous states

Partial Observability

All of the above

What is is a state space?

The **world state** includes every last detail of the environment



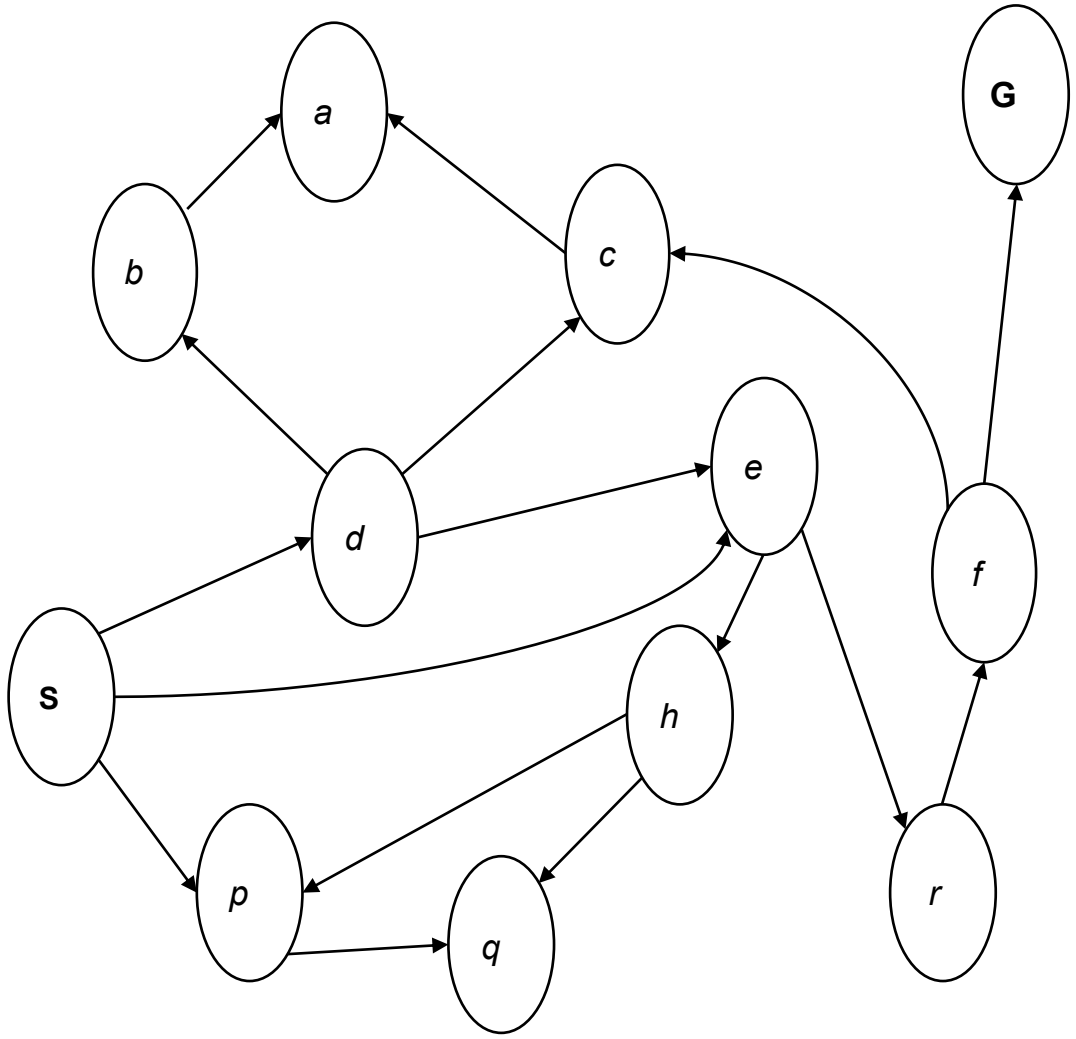
A **search state** keeps only the details needed for planning (abstraction)

- Problem: Pathing
  - States: (x,y) location
  - Actions: NSEW
  - Successor: update location only
  - Goal test: is (x,y)=END

- Problem: Eat-All-Dots
  - States: {(x,y), dot booleans}
  - Actions: NSEW
  - Successor: update location and possibly a dot boolean
  - Goal test: dots all false

Representing Search

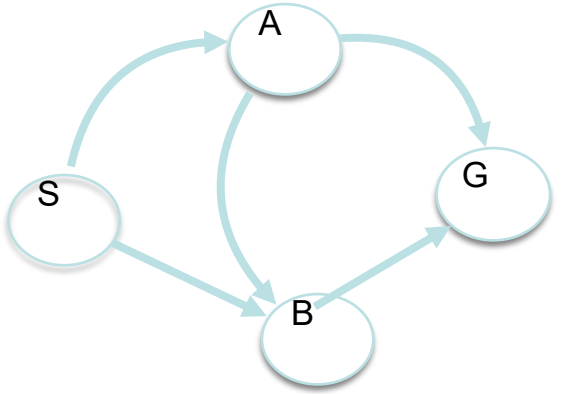
- **State space graph**
  - Vertices correspond to states (one vertex for each state)
  - Edges correspond to successors
  - Goal test is a set of goal nodes
- We search for a solution by building **a search tree** and traversing it to find a goal state

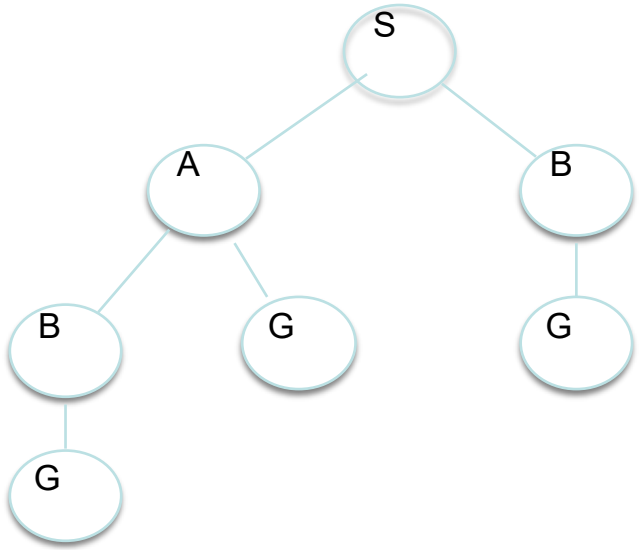


Search Tree

A search tree:

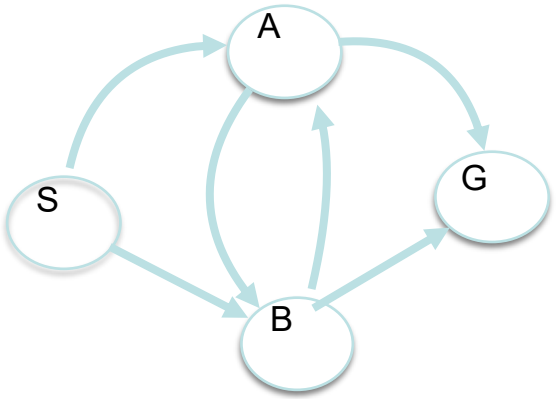
- Start state is the root of the tree
- Children are successors
- A plan is a path in the tree. A solution is a path from the root to a goal node.
- For most problems we do not actually generate the entire tree





Quiz

Given this state graph, how large is the search tree?



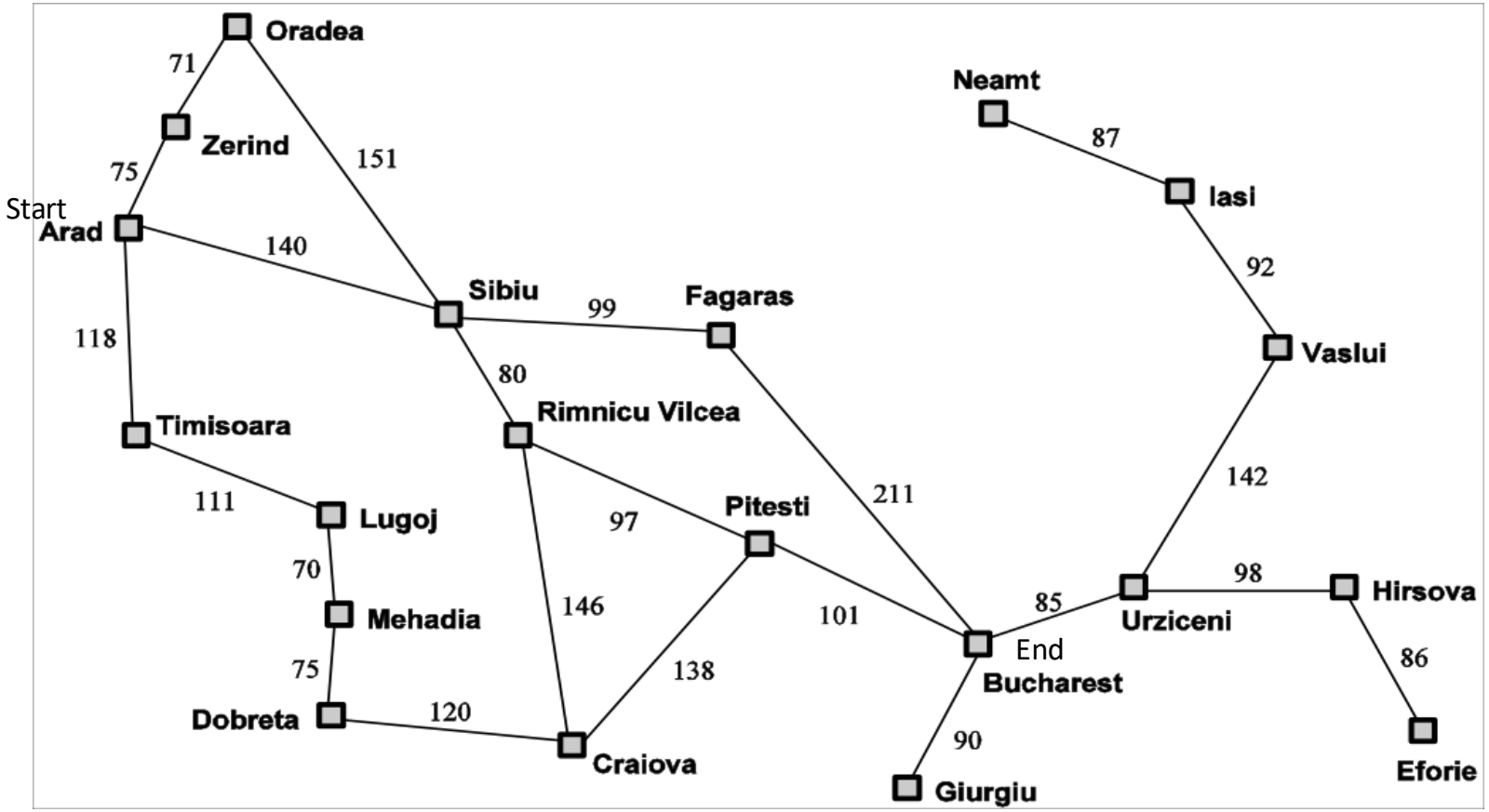
Expanding Nodes

Expanding a node:

Applying all legal operators to the state contained in the node

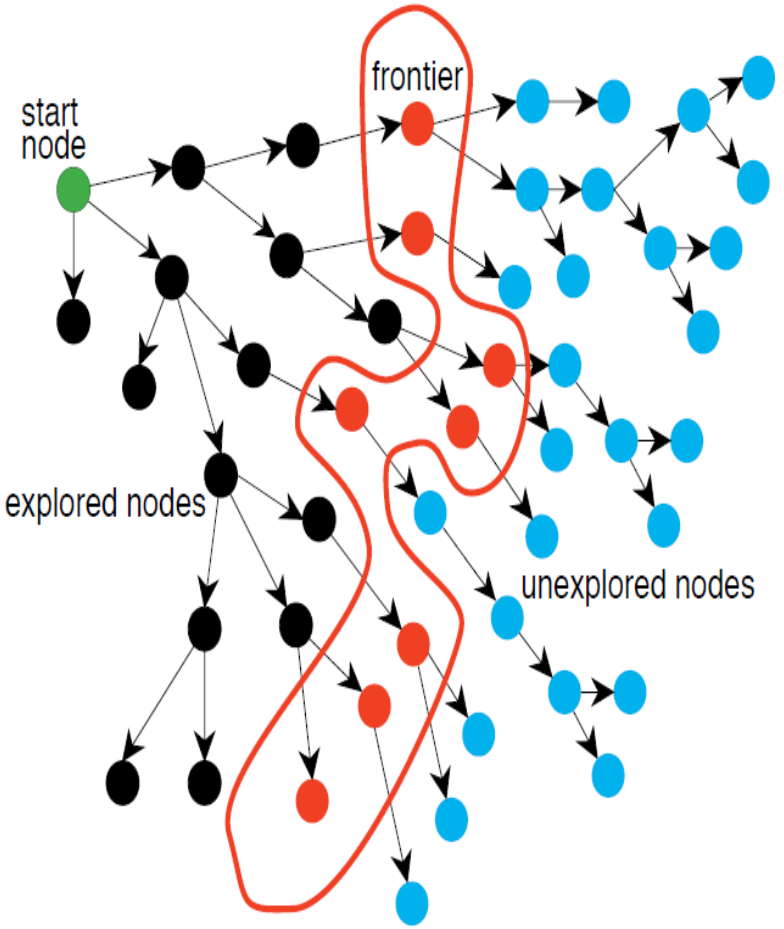
Generating nodes for all corresponding successor states

Example: Traveling in Romania



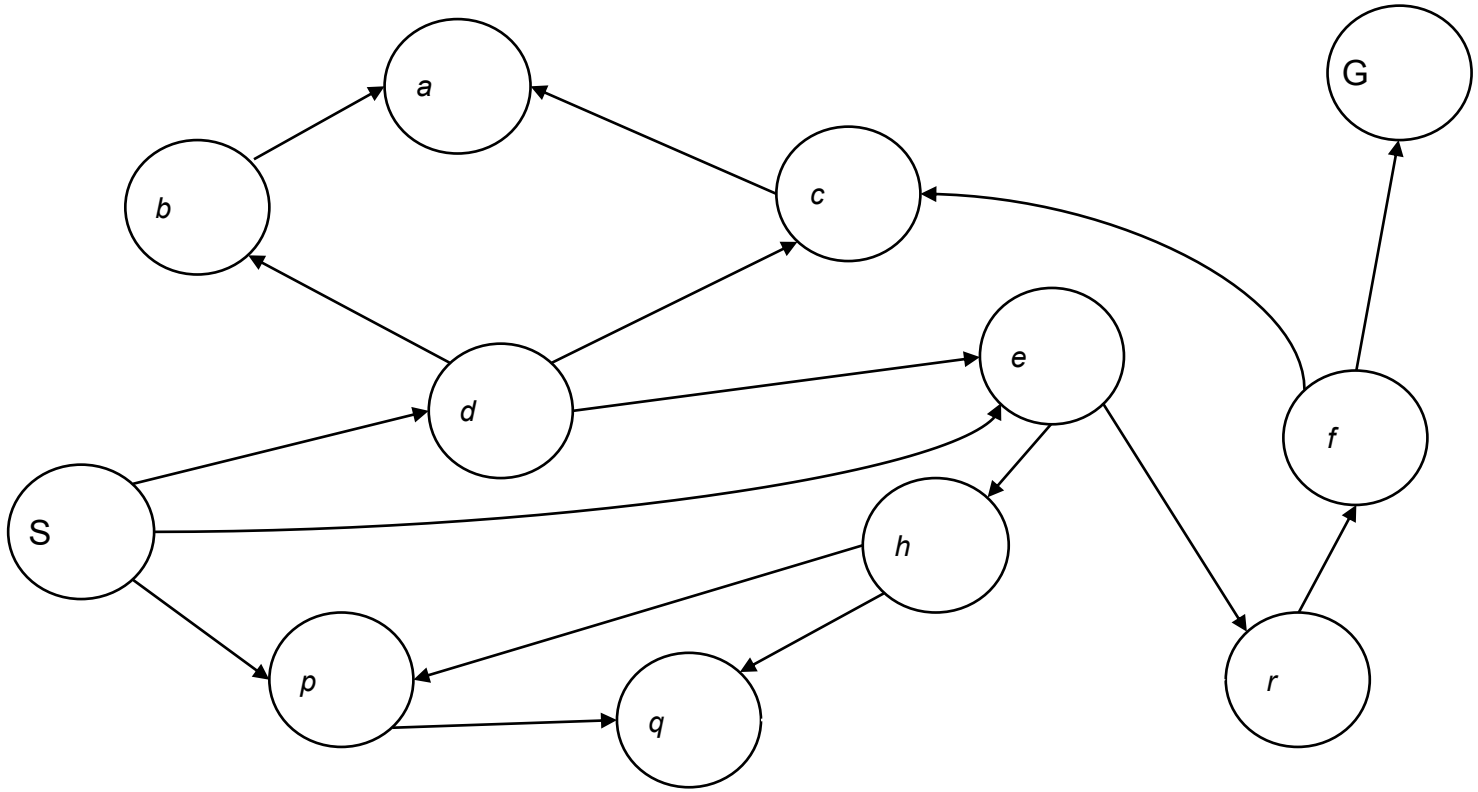
Generic Search Algorithm

- Initialize with initial state of the problem
- Repeat
  - If no candidate nodes can be expanded **return failure**
  - Choose leaf node for expansion, according to **search strategy**
  - If node contains goal state, **return solution**
  - Otherwise, expand the node. Add resulting nodes to the tree



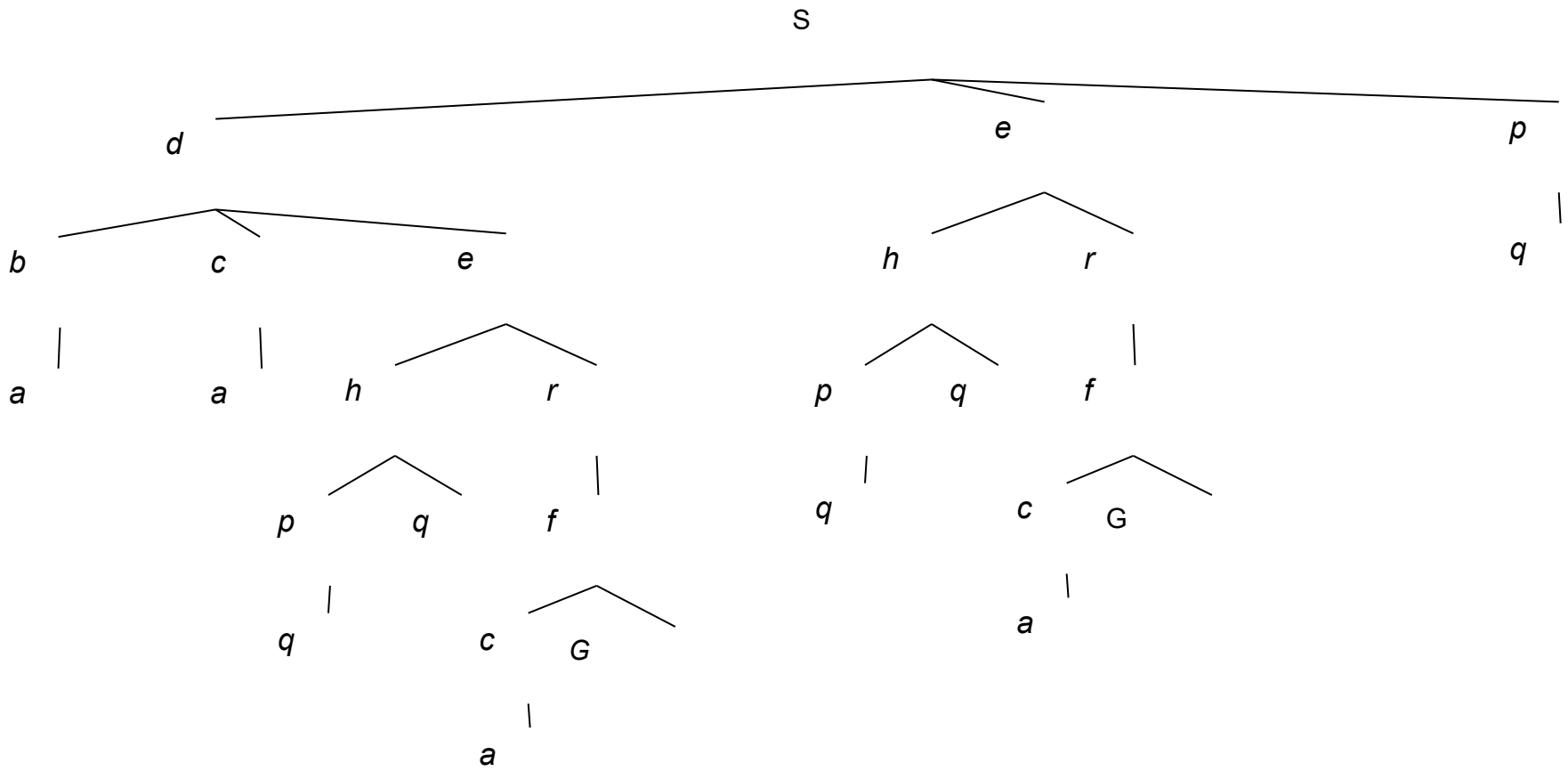
Implementation Details

Search Strategies



Adapted from UC Berkeley's CS188 Course

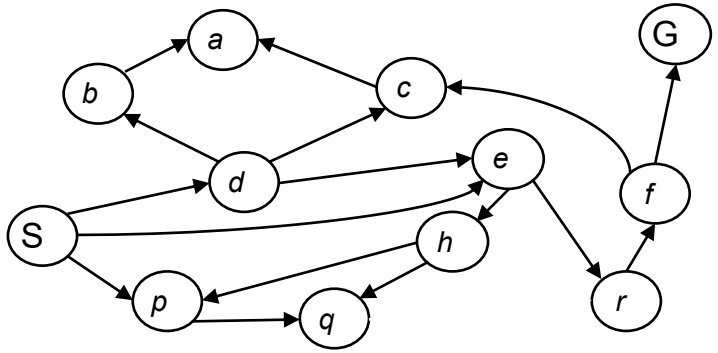
Search Strategies

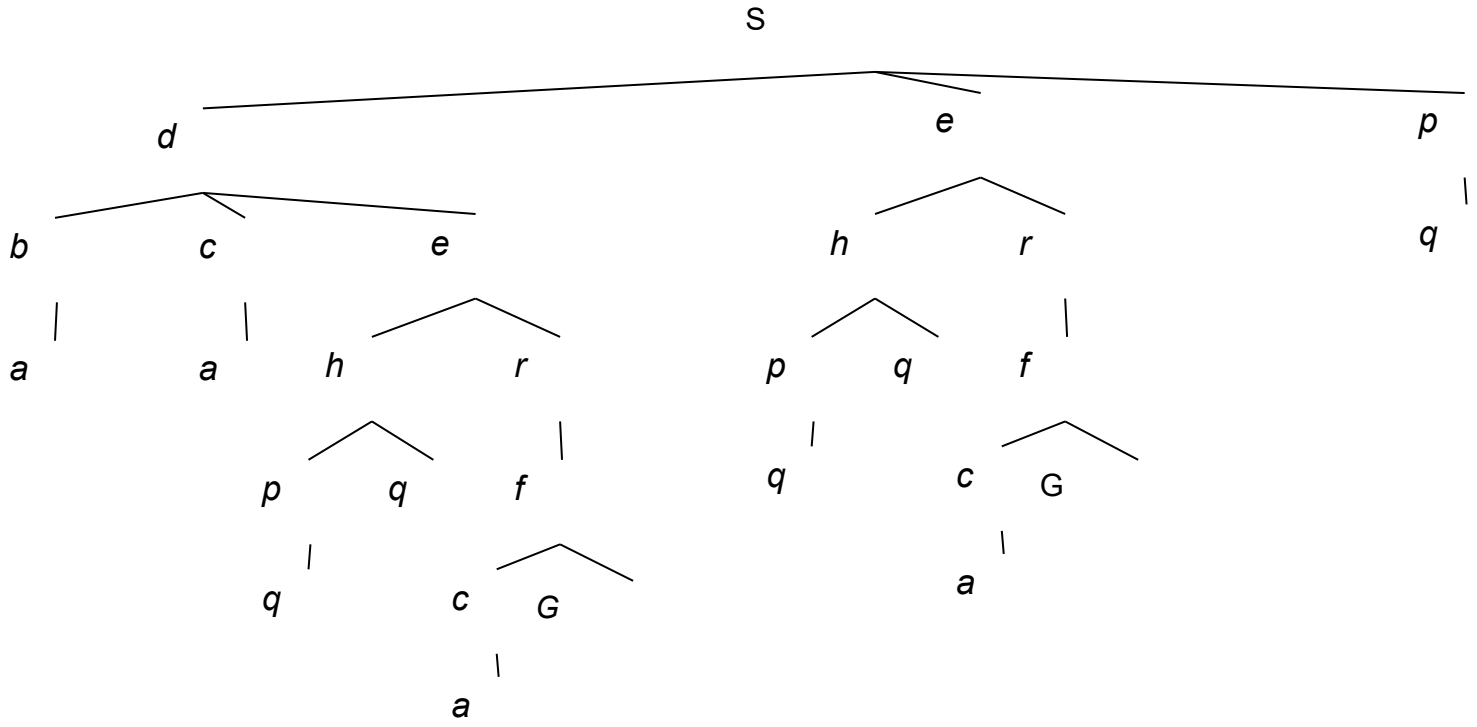


Depth-First Search

Strategy: Expand deepest node first

Implementation: LIFO stack

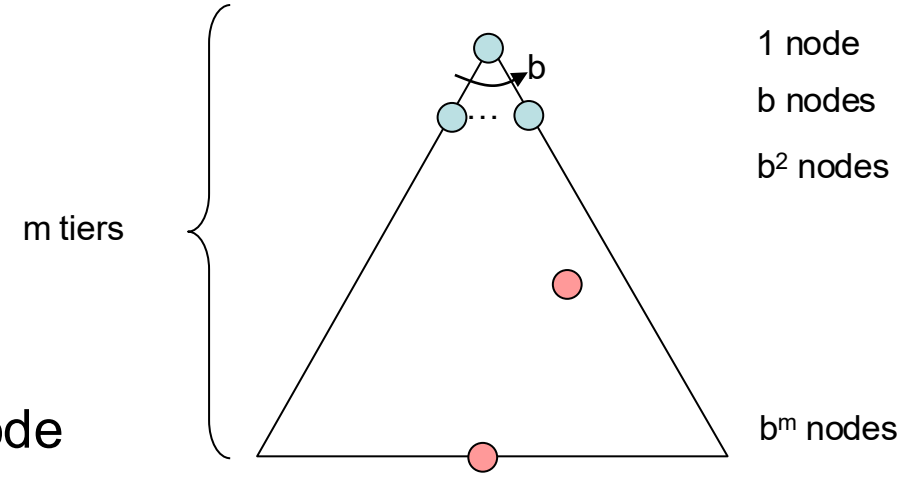




Key Properties

- **Completeness**: Is the alg. guaranteed to find a solution if the solution exists?
- **Optimality:** Does the alg. find the optimal solution?
- **Time complexity**
- **Space complexity** (size of the fringe)

b: branching factor
m: maximum depth
d: depth of shallowest goal node



Number of nodes in tree? $1+b+b^2+\ldots+b^m=O(b^m)$

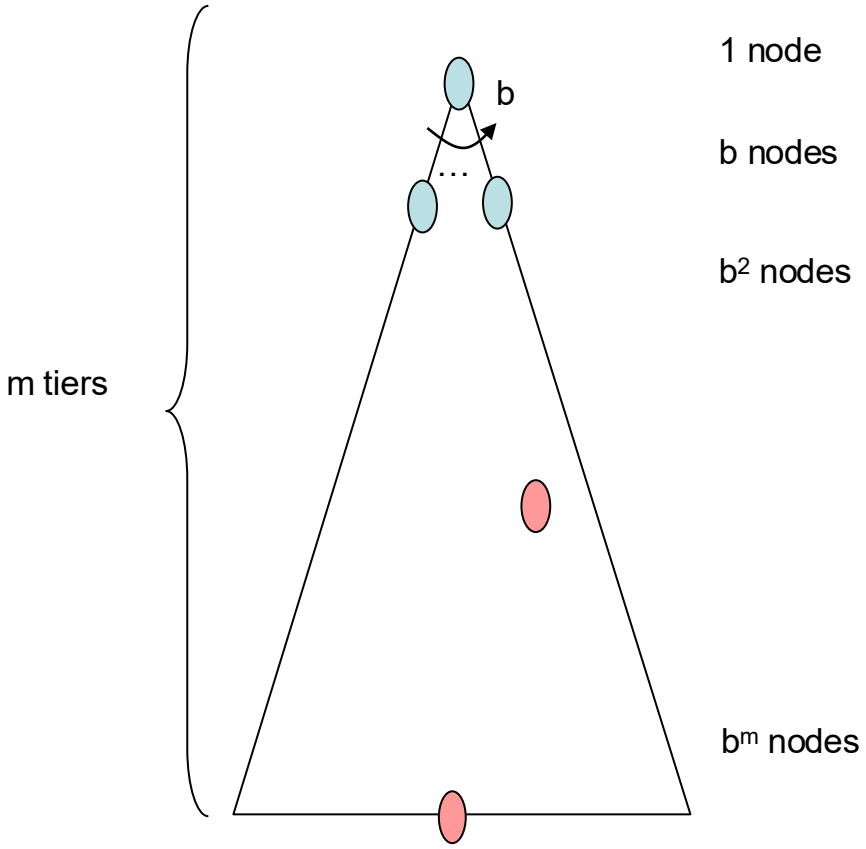
DFS Properties

Complete?

Optimal?

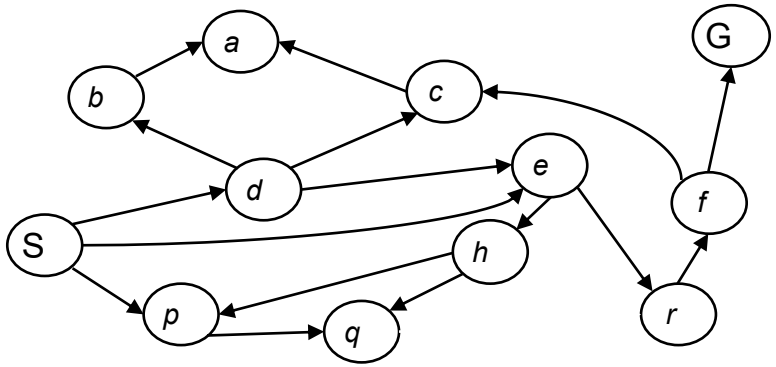
Time complexity

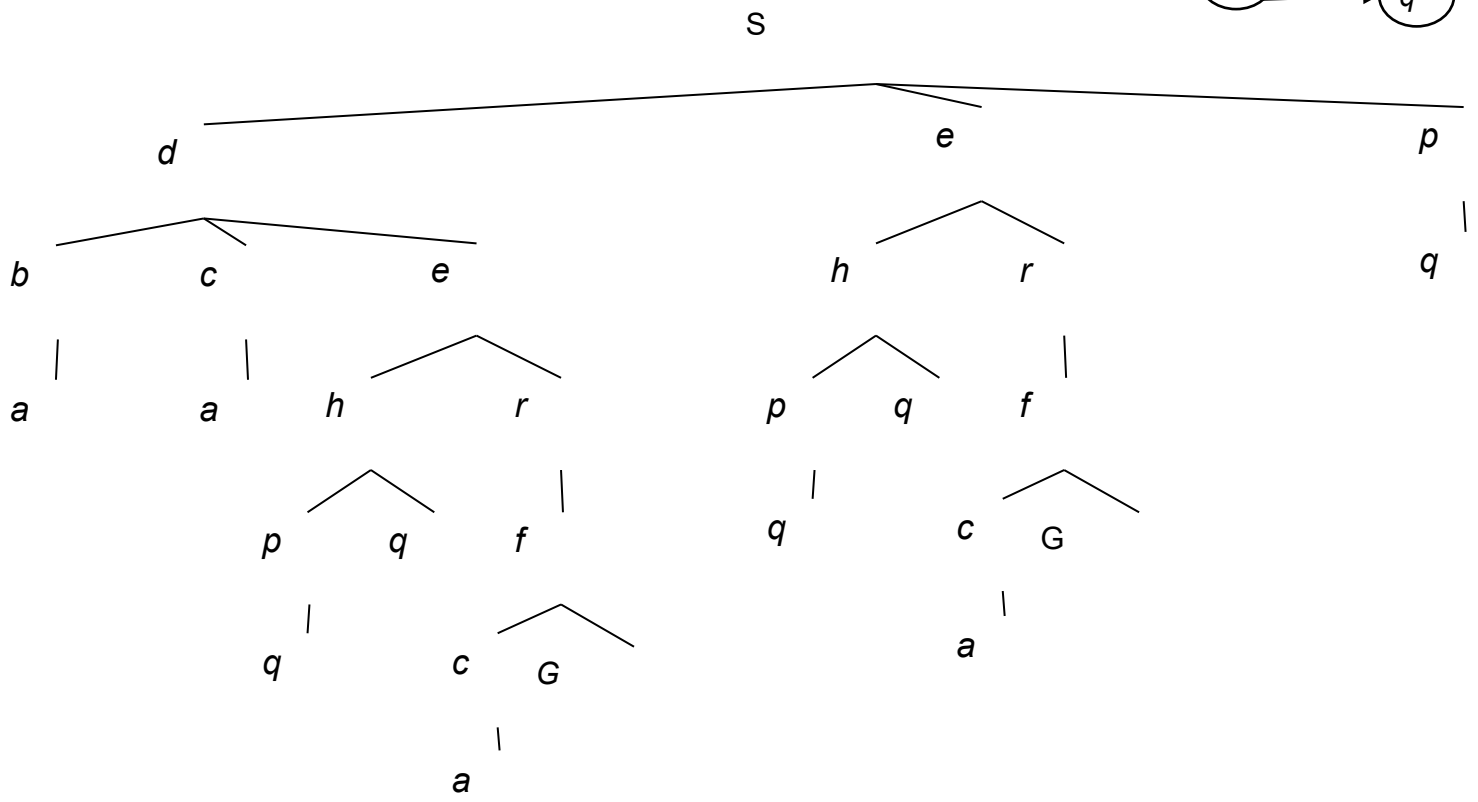
Space complexity



Breadth-First Search

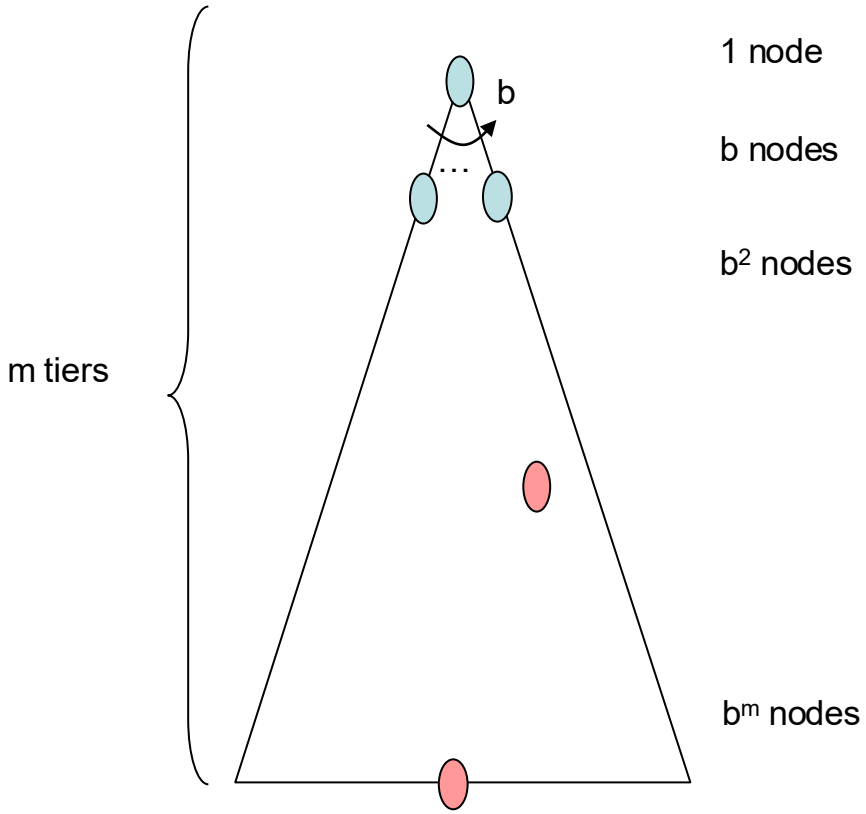
Strategy: Expand shallowest node first
Implementation: FIFO queue



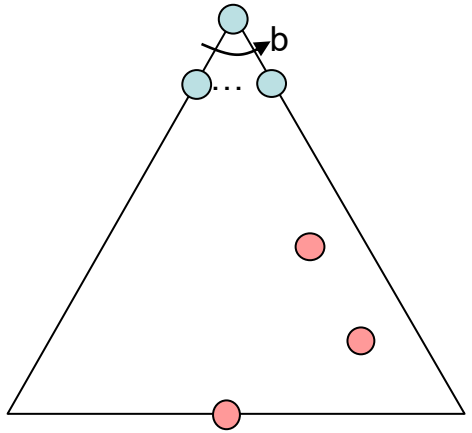


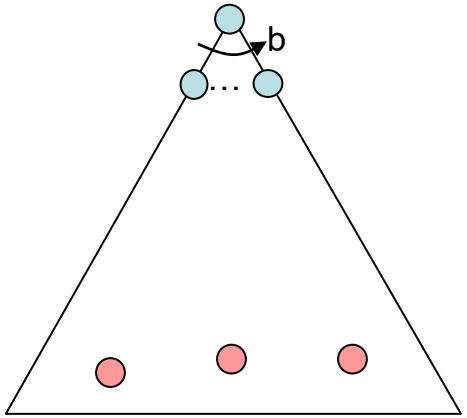
BFS Properties

- **Complete?**
- **Optimal?**
- **Time complexity**
- **Space complexity**



Quiz: DFS vs BFS





Iterative Deepening Search

- Can we combine search methods to take advantage of DFS space complexity and BFS completeness/shallow solution advantage?

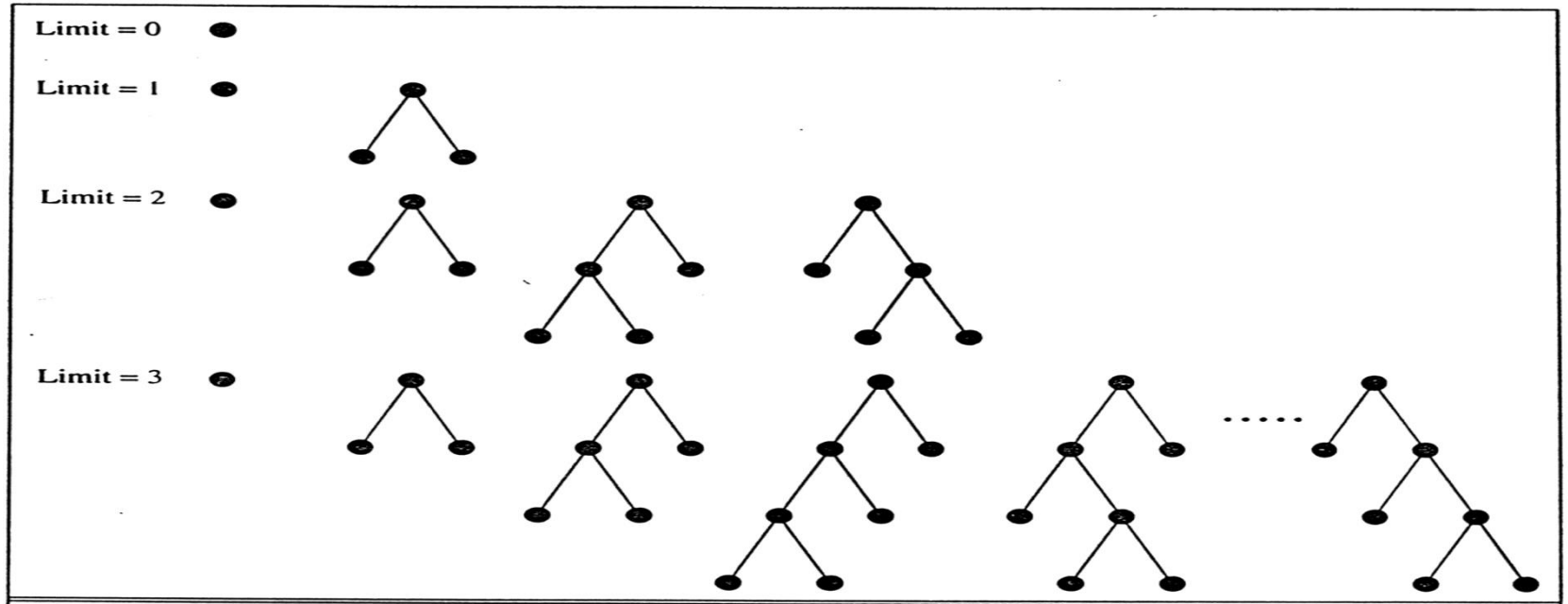
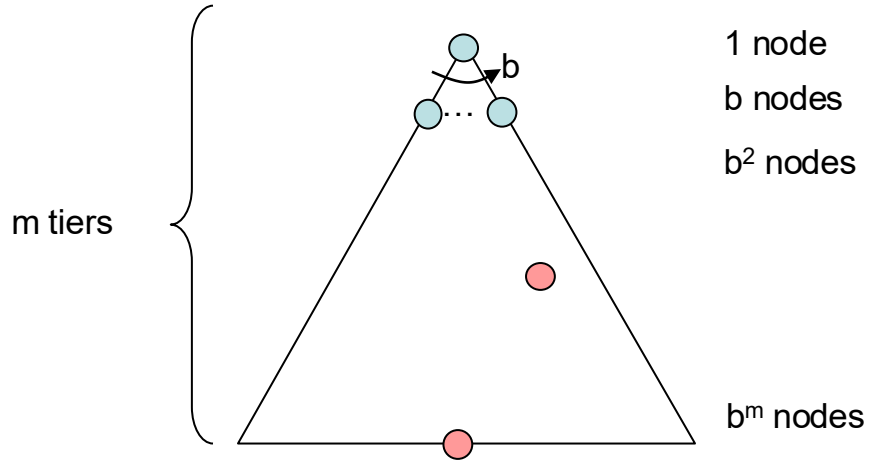


Figure 3.16 Four iterations of iterative deepening search on a binary tree.

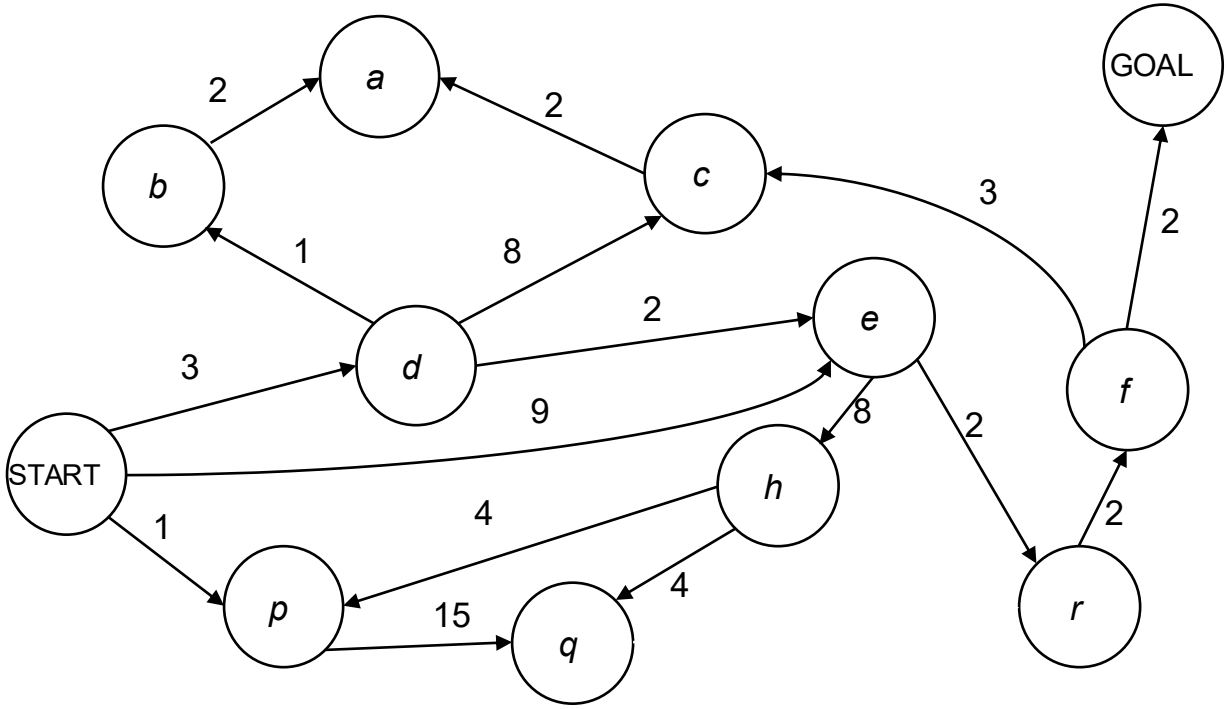
IDS Properties

- **Complete?**
- **Optimal?**
- **Time complexity**
- **Space complexity**



Wasteful? Most nodes found in lowest level of search so not too bad

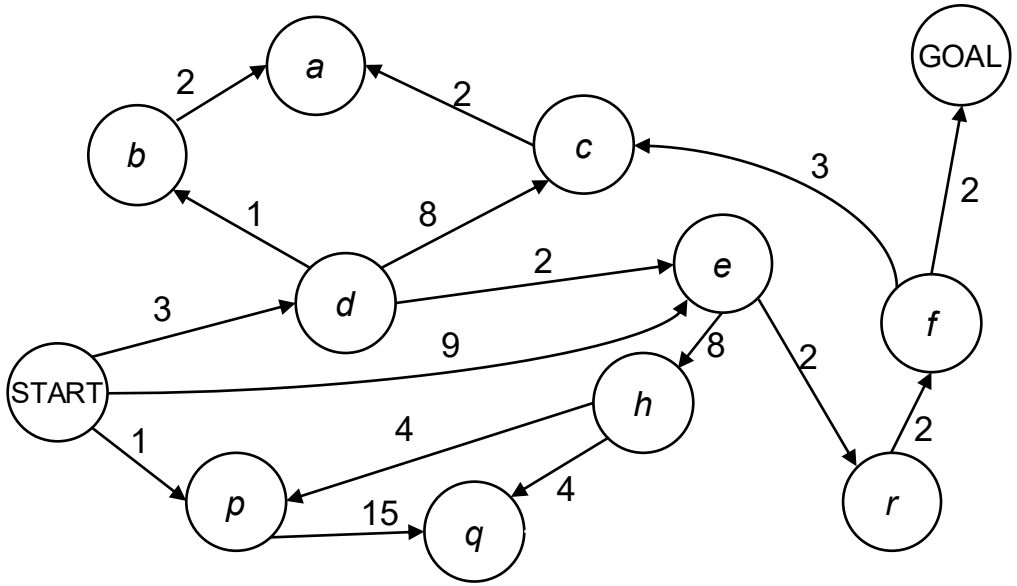
Cost-Sensitive Search



Recall that BFS was only optimal under some conditions (i.e. we only cared about number of actions taken). What can we do if actions have different costs?

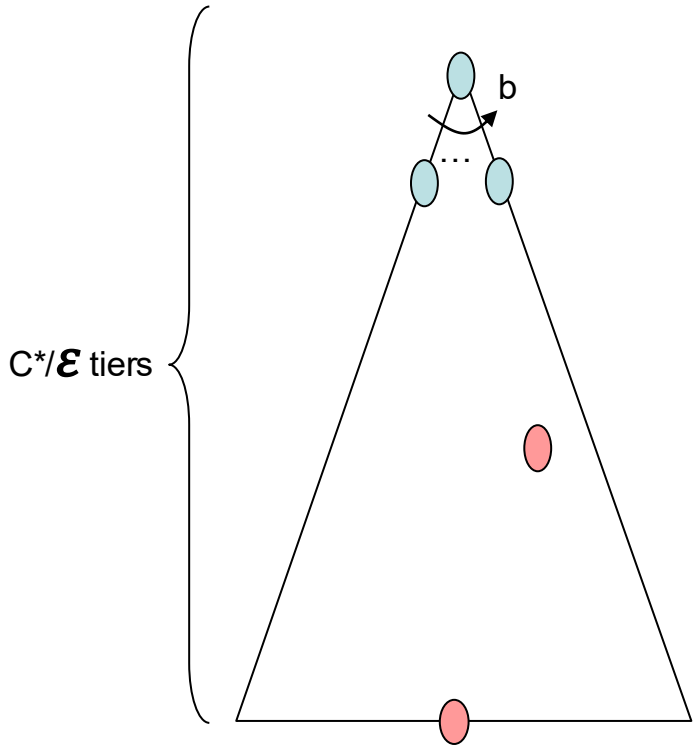
Uniform Cost Search

Strategy: Expand cheapest node first
Implementation: Priority queue



UCS Properties

- **Complete?**
- **Optimal?**
- **Time complexity**
- **Space complexity**



Summary

- These algorithms are basically the same except for the order in which they expand nodes
  - Basically all priority queues with different ways to determining priorities
- How successful the search is depends heavily on your model!

Criteria	BFS	DFS	IDS	UCS
Complete	Yes	No	Yes	Yes
Time	$O(b^d)$	$O(b^m)$	$O(b^d)$	$O(b^{ceiling(C^*/\varepsilon)})$
Space	$O(b^d)$	O(bm)	O(bd)	$O(b^{ceiling(C^*/\varepsilon)})$
Optimal	Yes	No	Yes	Yes