CSE 473: Artificial Intelligence
Spring 2014

Hanna Hajishirzi

Search with Cost & Heuristics

slides from
Dan Klein, Stuart Russell, Andrew Moore, Dan Weld, Pieter Abbeel, Luke Zettelmoyer

Announcement

- PS1 will be on the web soon!
  - Start early

Recap: Search

- Search problem:
  - States (configurations of the world)
  - Successor function; drawn as a graph
  - Start state and goal test

- Search tree:
  - Nodes: represent plans for reaching states
  - Plans have costs (sum of action costs)

General Tree Search

- Search Algorithms:
  - Systematically builds a search tree
  - Chooses an ordering of the fringe (unexplored nodes)
- Important ideas:
  - Fringe
  - Expansion
  - Exploration strategy
- Main question: which fringe nodes to explore?

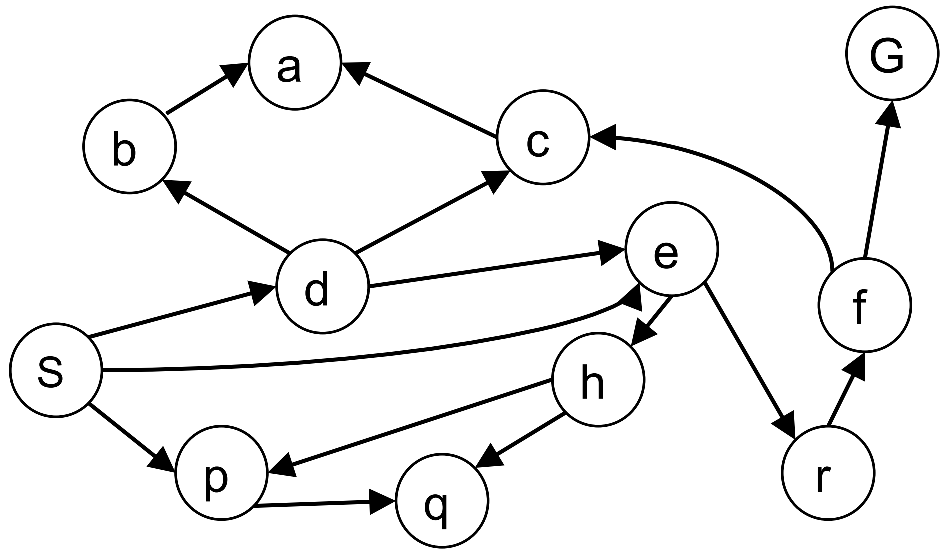
Outline

- Uninformed Search Methods (part review for some)
  - Depth-First Search
  - Breadth-First Search
  - Uniform-Cost Search
- Heuristic Search Methods (new for all)
  - Best First / Greedy Search
  - A*

Review: Depth First Search

Strategy: expand deepest node first

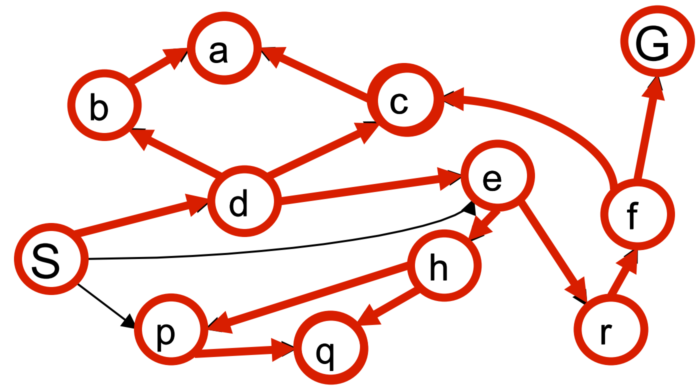
Implementation: Fringe is a LIFO queue (a stack)

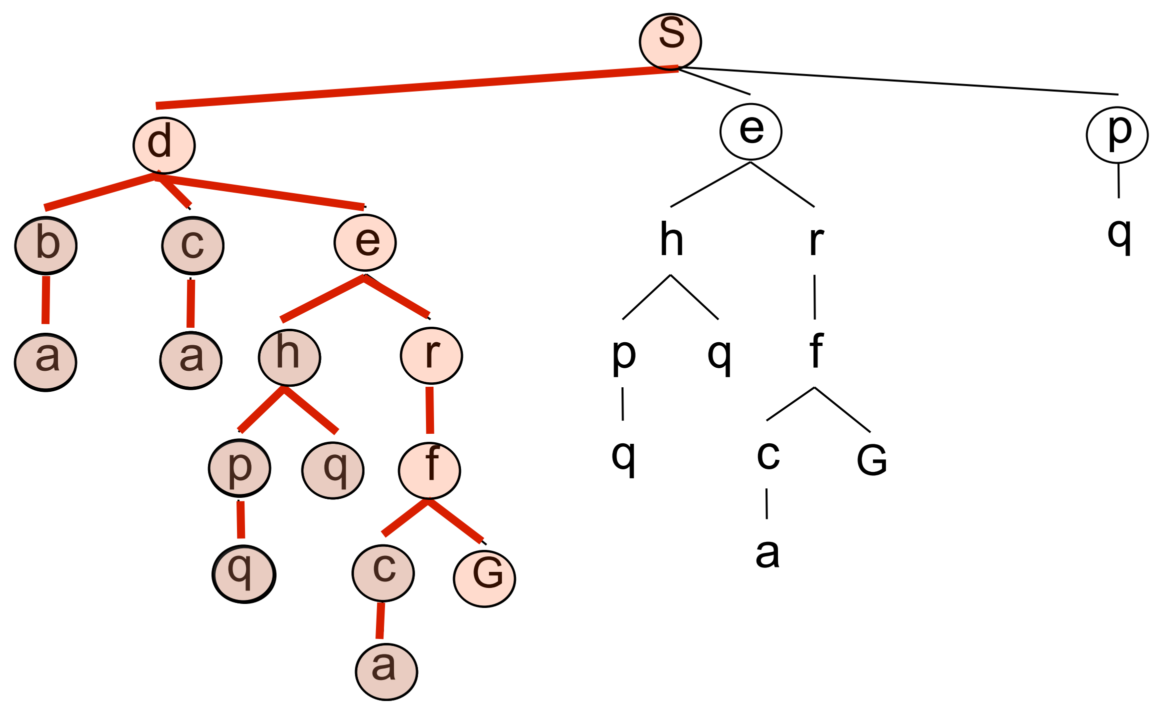


Review: Depth First Search

Expansion ordering:

(d,b,a,c,a,e,h,p,q,q,r,f,c,a,G)

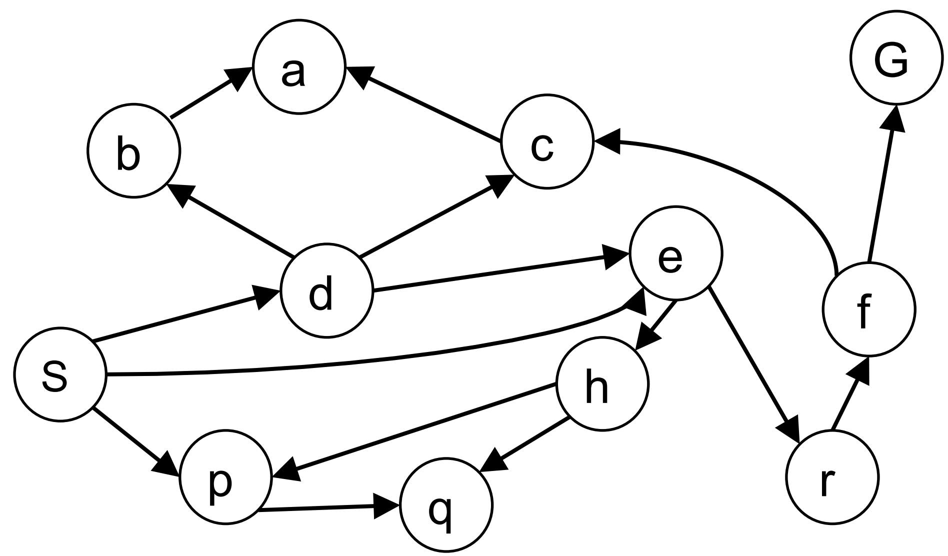




Review: Breadth First Search

Strategy: expand shallowest node first

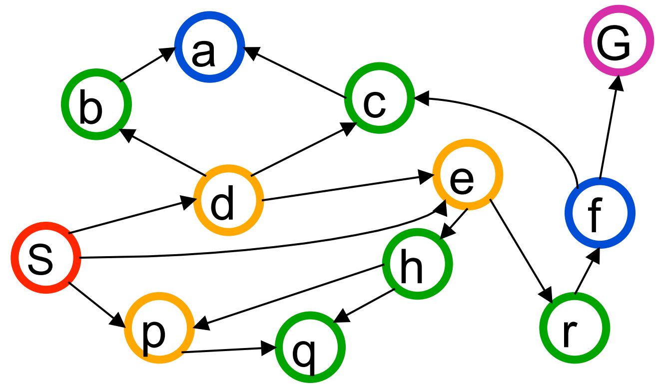
Implementation: Fringe is a FIFO queue



Review: Breadth First Search

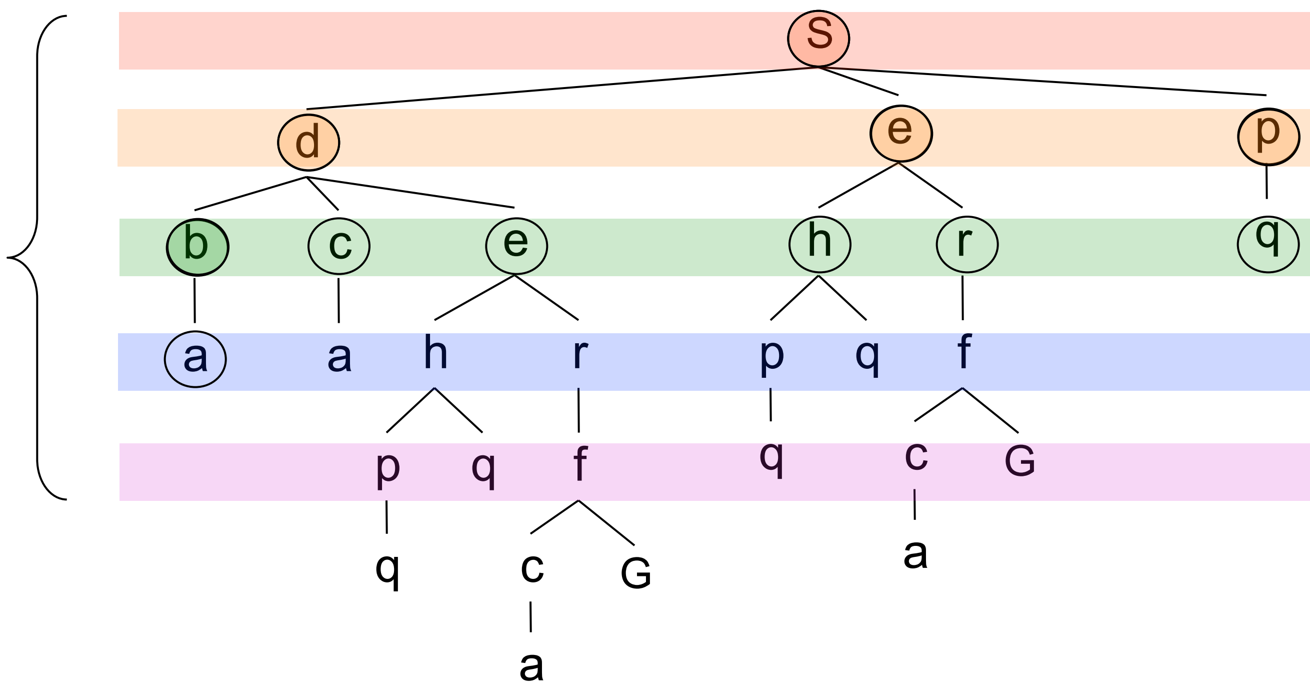
Expansion order:

(S,d,e,p,b,c,e,h,r,q,a,a
,h,r,p,q,f,p,q,f,q,c,G)



Search

Tiers



Search Algorithm Properties

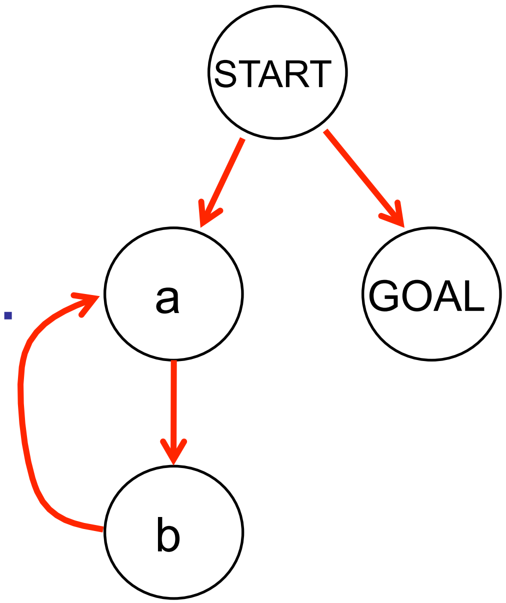
- Complete? Guaranteed to find a solution if one exists?
- Optimal? Guaranteed to find the least cost path?
- Time complexity?
- Space complexity?

Variables:

n	Number of states in the problem
b	The maximum branching factor B (the maximum number of successors for a state)
C^*	Cost of least cost solution
d	Depth of the shallowest solution
m	Max depth of the search tree

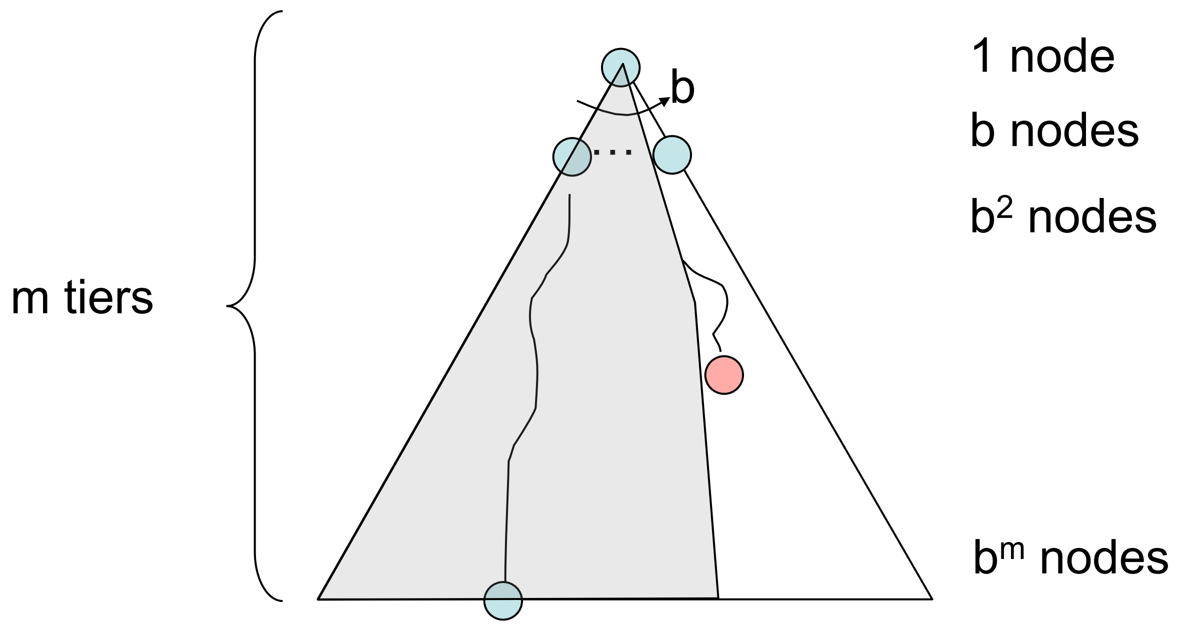
DFS

Algorithm		Complete	Optimal	Time	Space
DFS	Depth First Search	No	No	Infinite	Infinite



- Infinite paths make DFS incomplete…
  - How can we fix this?
  - Check new nodes against path from S
- Infinite search spaces still a problem
  - If the left subtree has unbounded depth

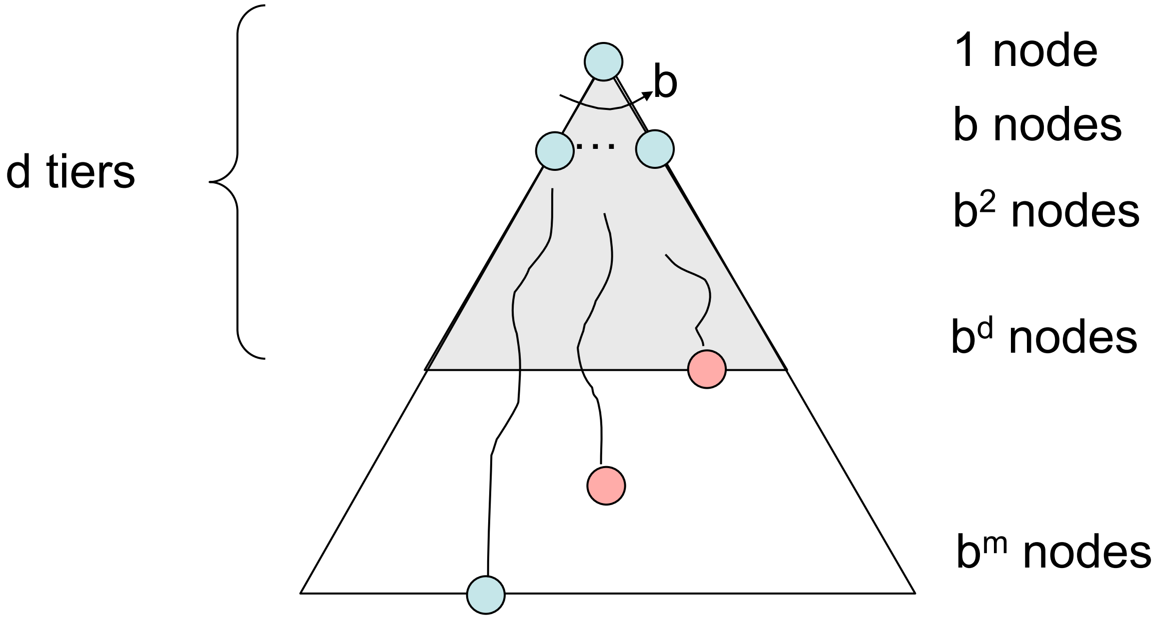
DFS



Algorithm		Complete	Optimal	Time	Space
DFS	w/ Path Checking	Y if finite	N	$O(b^m)$	$O(bm)$

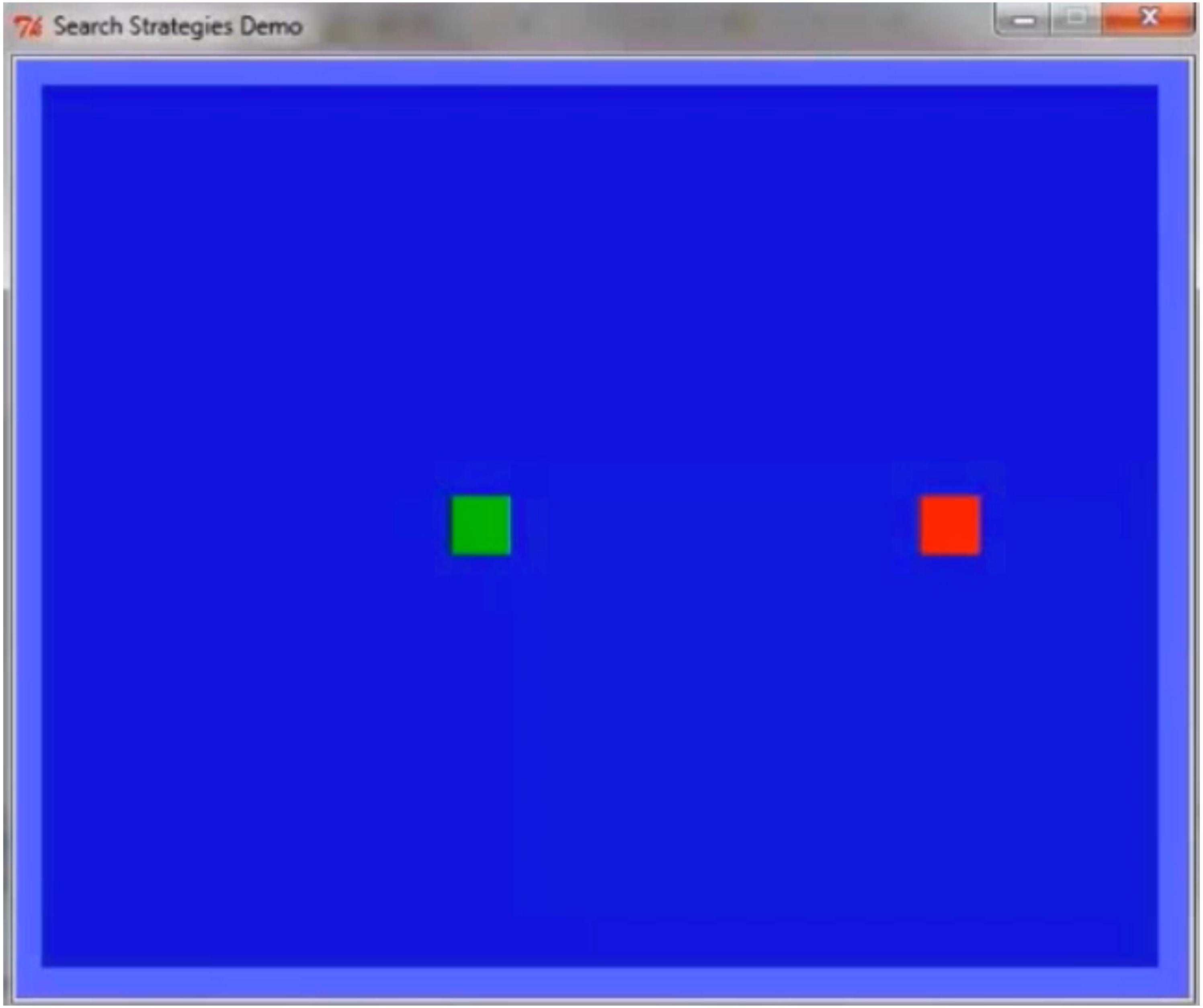
BFS

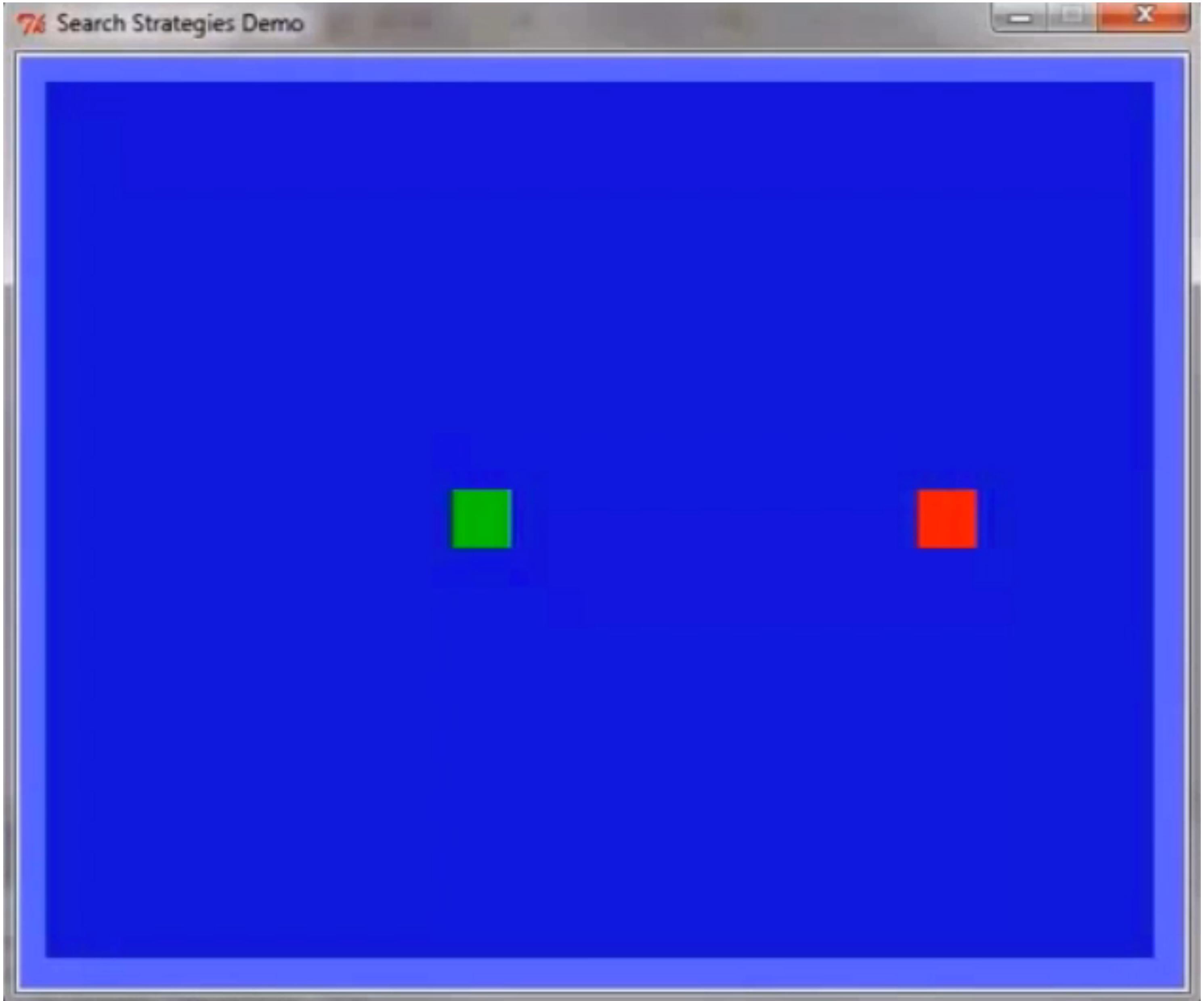
Algorithm		Complete	Optimal	Time	Space
DFS	w/ Path Checking	Y	N	$O(b^m)$	$O(bm)$
BFS		Y	Y*	$O(b^d)$	$O(b^d)$



Comparisons

- When will BFS outperform DFS?
- When will DFS outperform BFS?


Search Strategies Demo

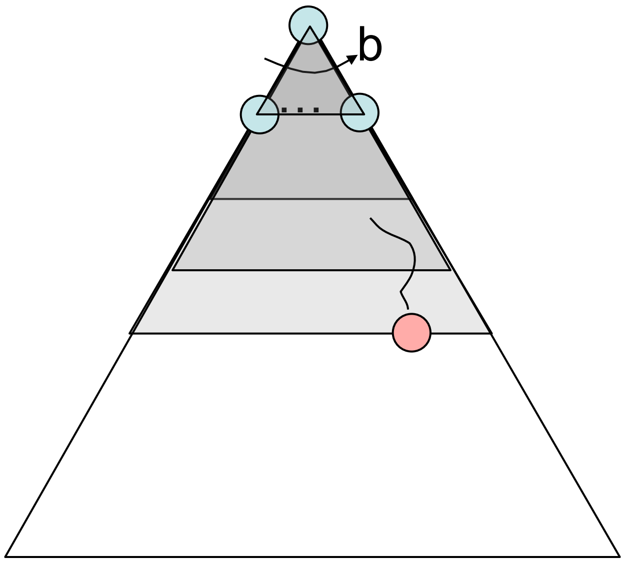

Search Strategies Demo

Iterative Deepening

Iterative deepening uses DFS as a subroutine:

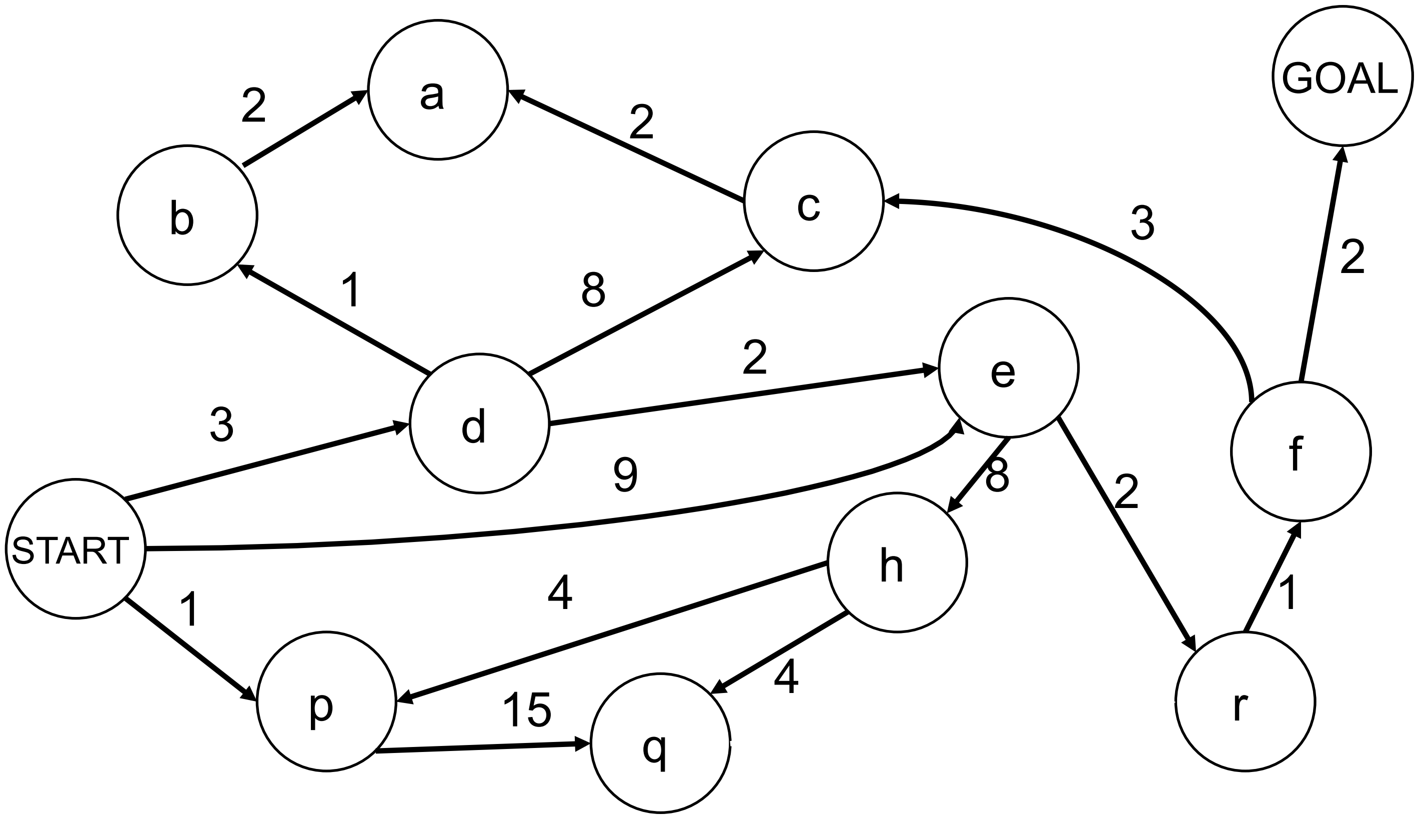
1. Do a DFS which only searches for paths of length 1 or less.
2. If "1" failed, do a DFS which only searches paths of length 2 or less.
3. If "2" failed, do a DFS which only searches paths of length 3 or less.

….and so on.



Algorithm		Complete	Optimal	Time	Space
DFS	w/ Path Checking	Y	N	$O(b^m)$	$O(bm)$
BFS		Y	Y*	$O(b^d)$	$O(b^d)$
ID		Y	Y*	$O(b^d)$	$O(bd)$

Costs on Actions

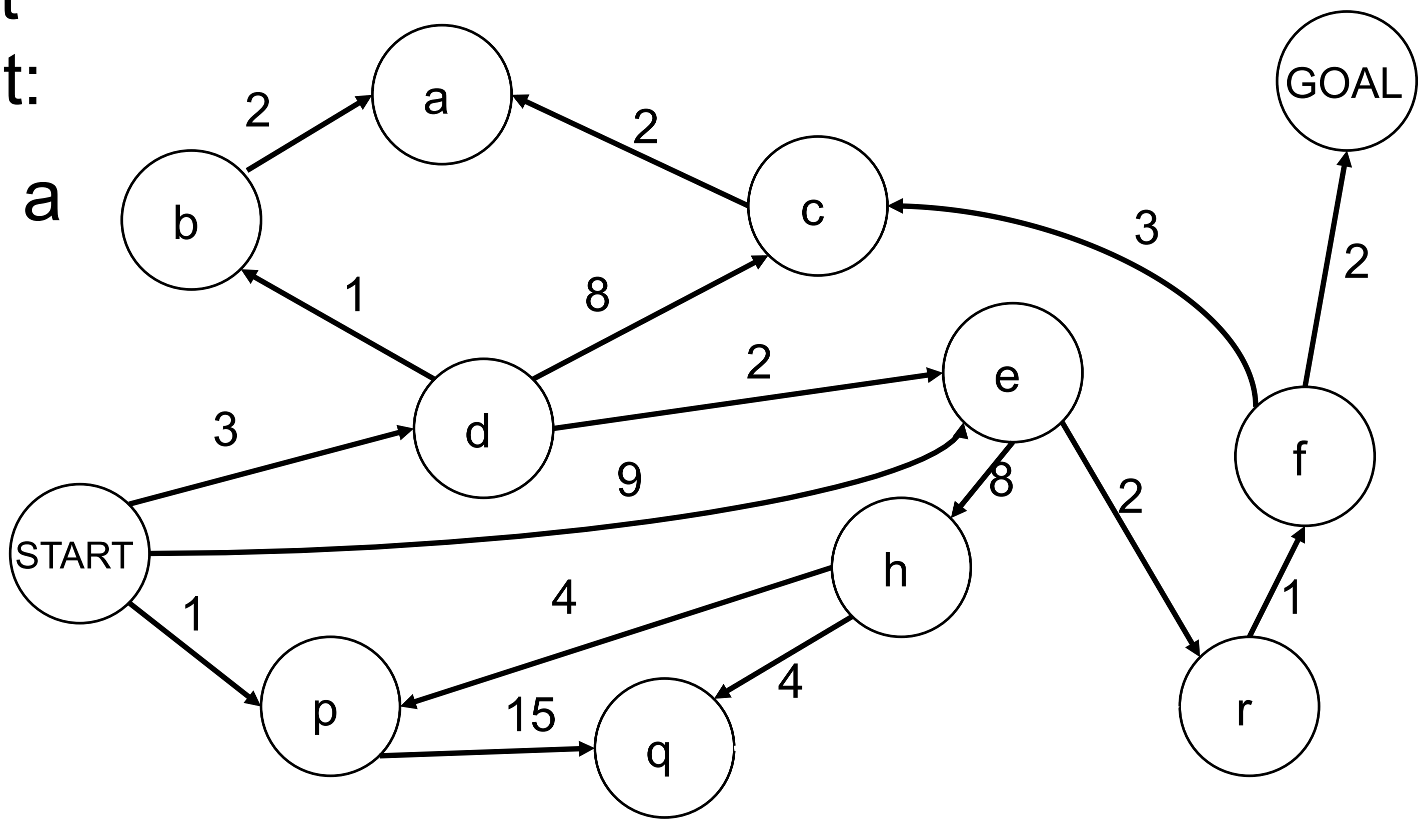


Notice that BFS finds the shortest path in terms of number of transitions. It does not find the least-cost path.

Uniform Cost Search

Expand cheapest node first:

Fringe is a priority queue



Uniform Cost Search

- Generalization of breadth-first search
- ***Priority*** queue of nodes to be explored
- Cost function f(n) applied to each node

```
Add initial state to priority queue
While queue not empty
        Node = head(queue)
        If goal?(node) then return node
        Add children of node to queue
```

Priority Queue Refresher

- A priority queue is a data structure in which you can insert and retrieve (key, value) pairs with the following operations:

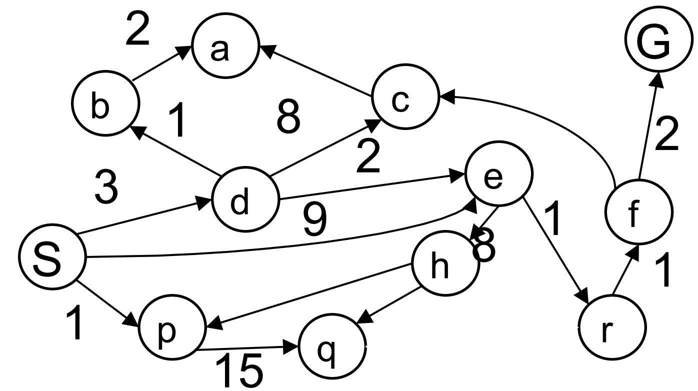
pq.push(key, value)	inserts (key, value) into the queue.
pq.pop()	returns the key with the lowest value, and removes it from the queue.

- Unlike a regular queue, insertions aren't constant time, usually O(log n)
- We'll need priority queues for cost-sensitive search methods

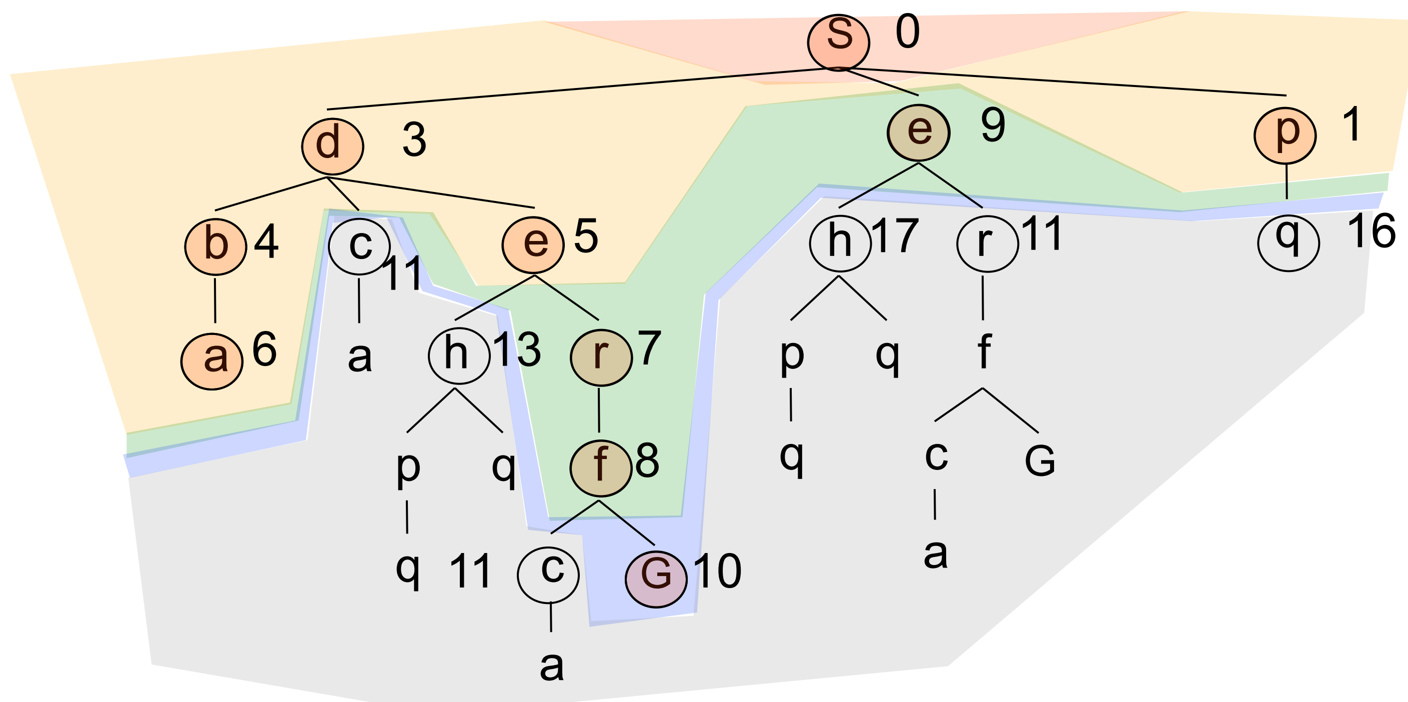
Uniform Cost Search

Expansion order:

(S,p,d,b,e,a,r,f,e,G)

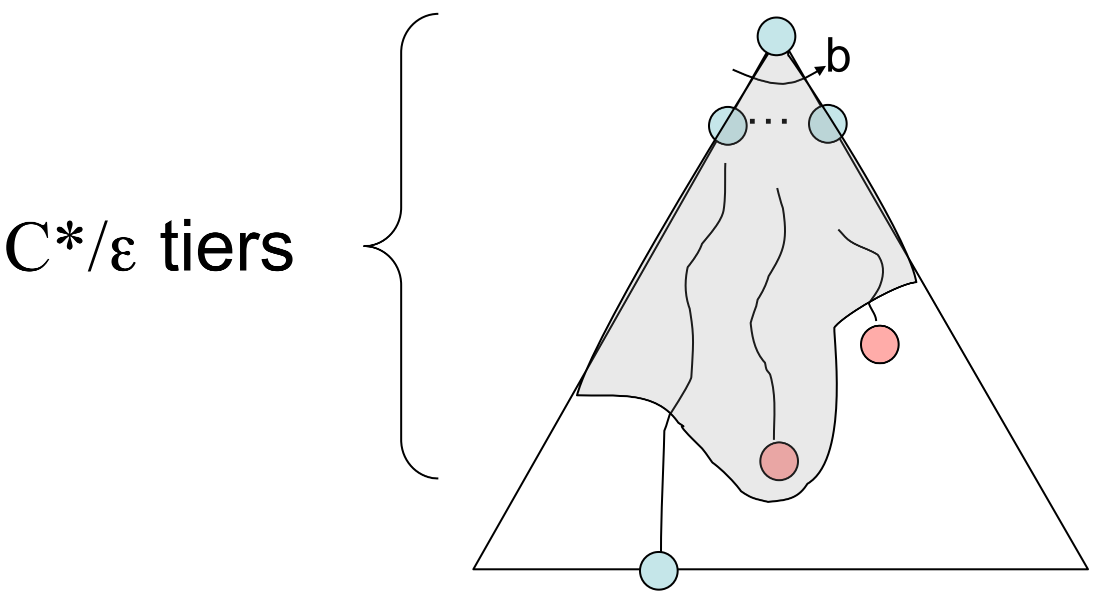


Cost contours



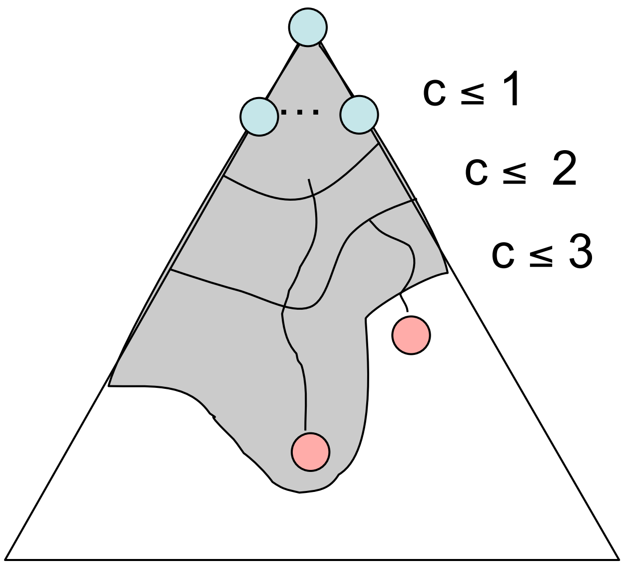
Uniform Cost Search

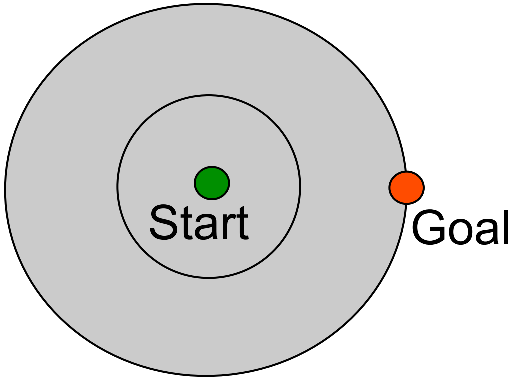
Algorithm		Complete	Optimal	Time	Space
DFS	w/ Path Checking	Y	N	$O(b^m)$	$O(bm)$
BFS		Y	Y*	$O(b^d)$	$O(b^d)$
UCS		Y*	Y	$O(b^{C*/\varepsilon})$	$O(b^{C*/\varepsilon})$



Uniform Cost Issues

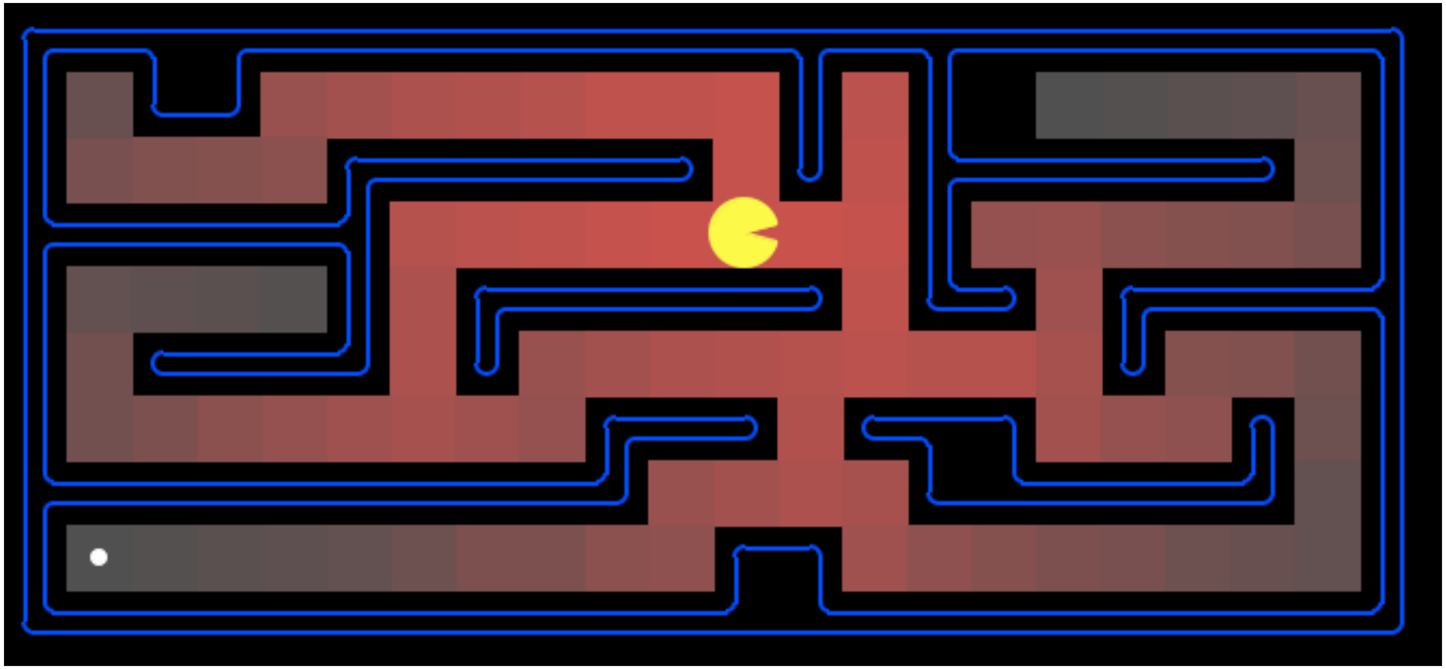
- Remember: explores increasing cost contours
- The good: UCS is complete and optimal!
- The bad:
  - Explores options in every "direction"
  - No information about goal location





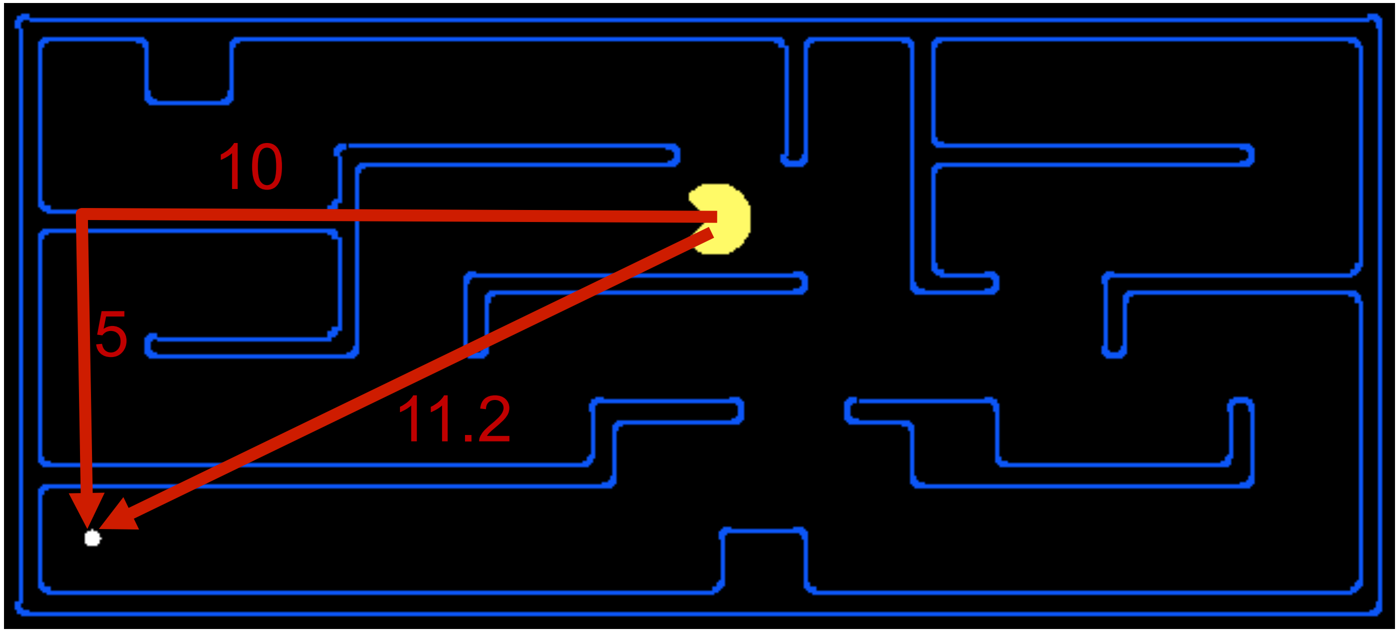
Uniform Cost: Pac-Man

- Cost of 1 for each action
- Explores all of the states, but one



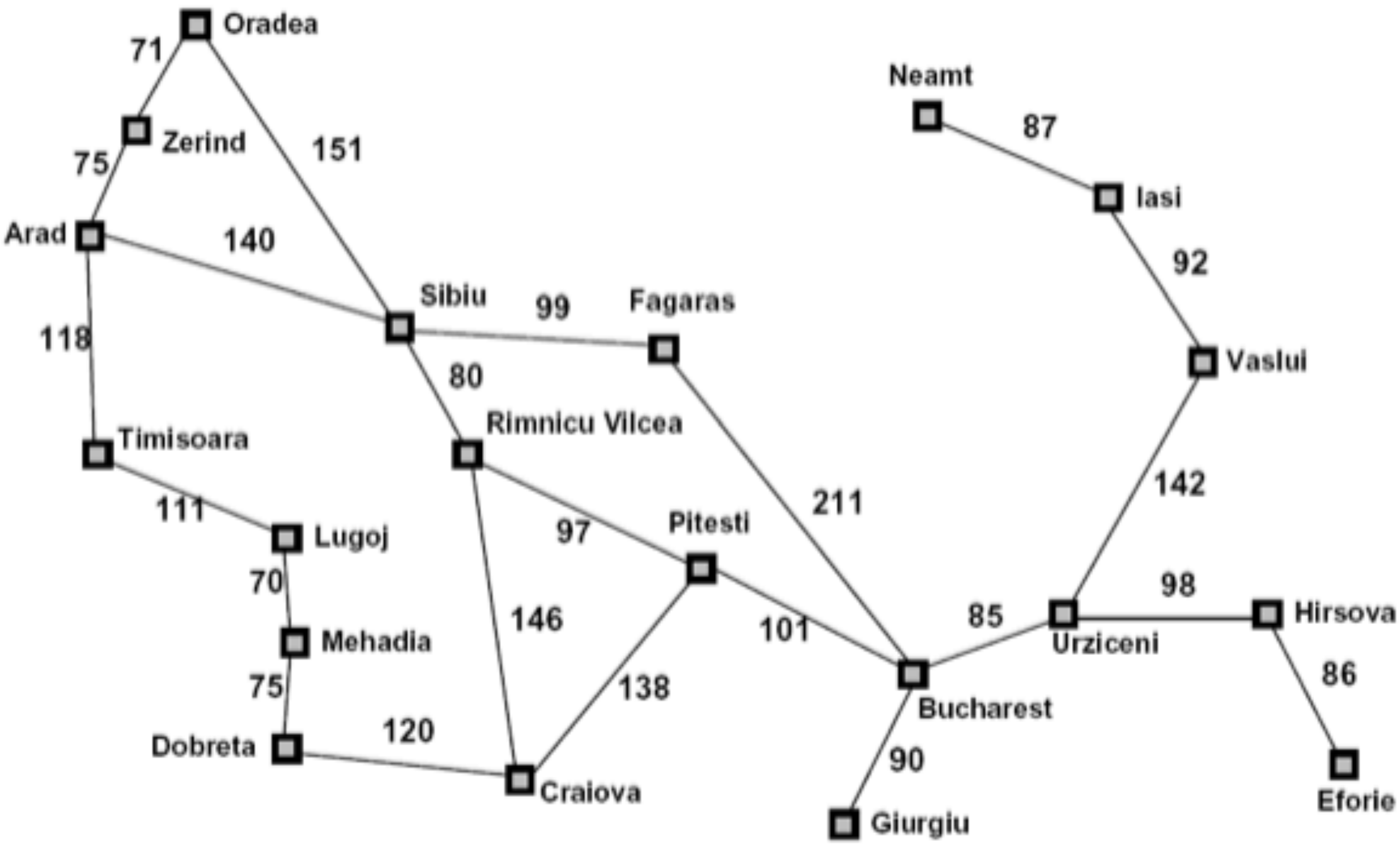
Search Heuristics

- Any estimate of how close a state is to a goal
- Designed for a particular search problem



- Examples: Manhattan distance, Euclidean distance

Heuristics



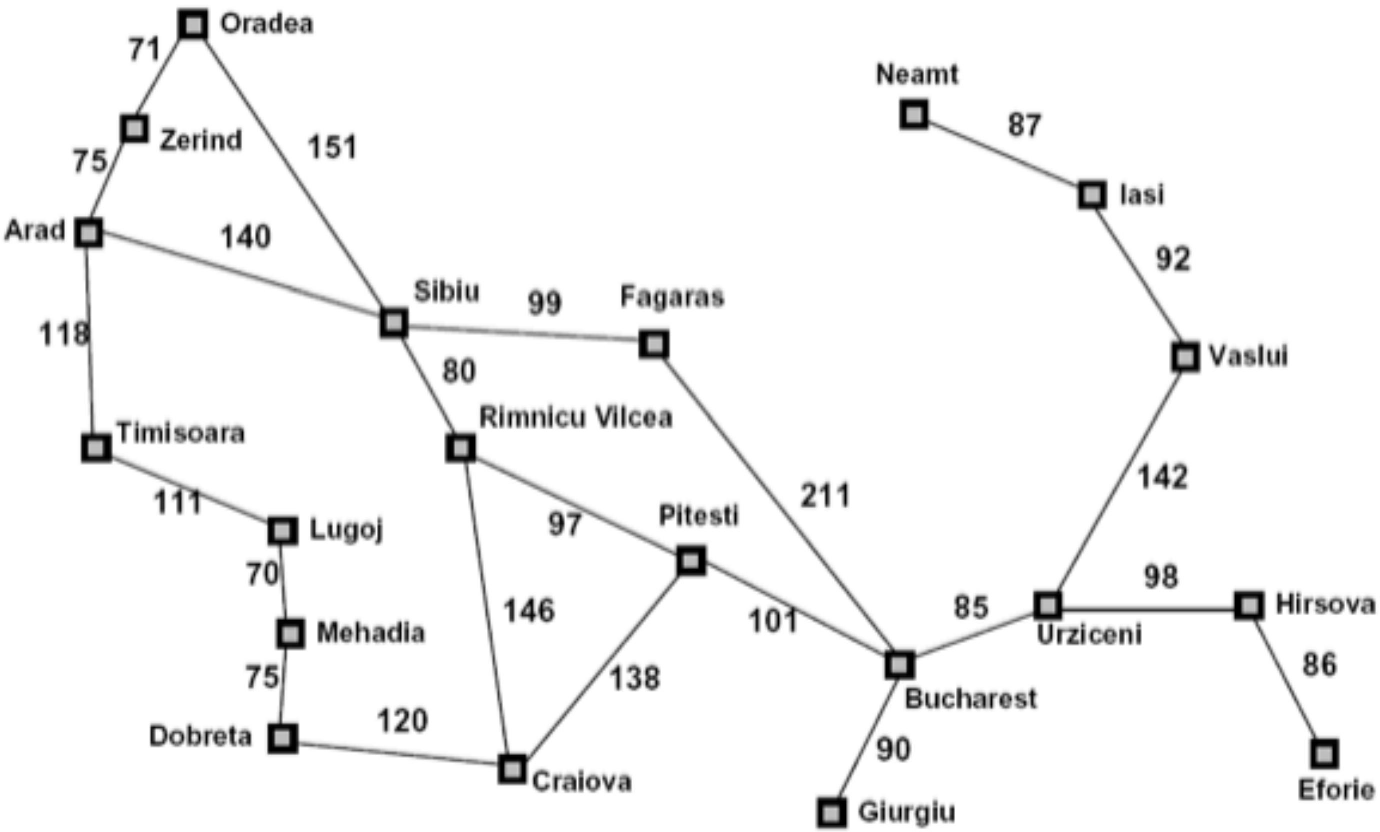
Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

H(x)

Best First / Greedy Search

Best first with f(n) = heuristic estimate of distance to goal

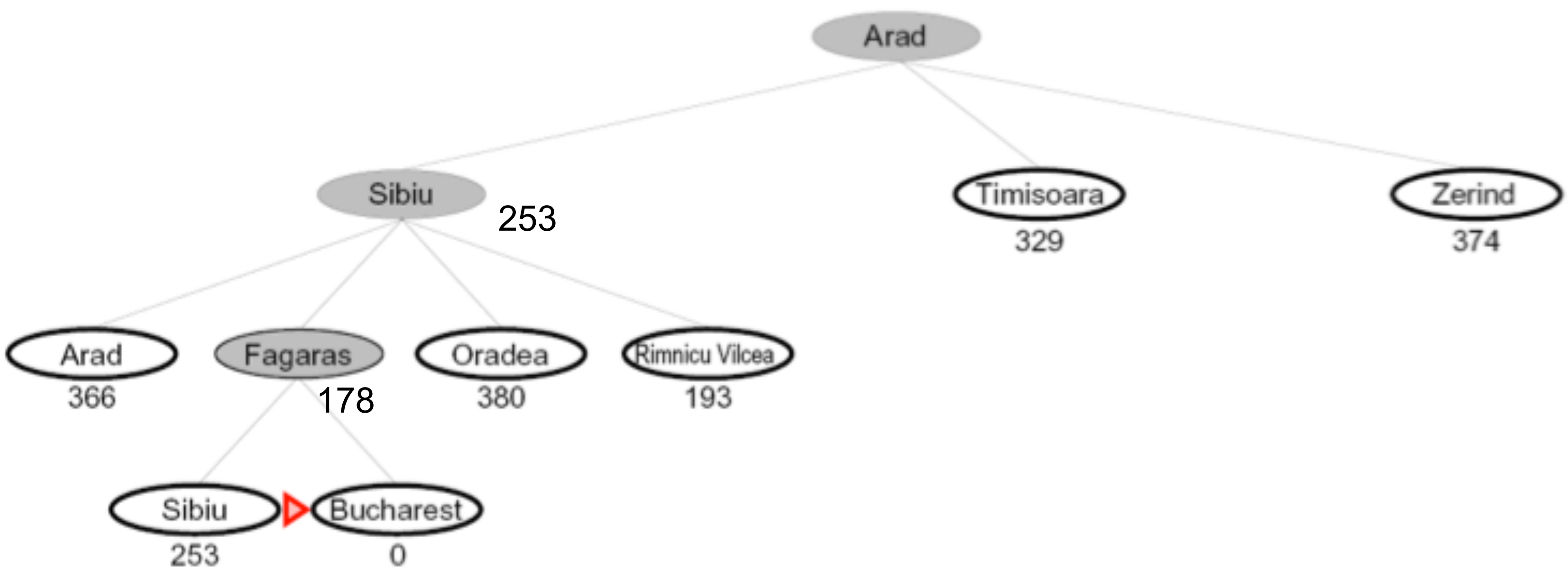


Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Best First / Greedy Search

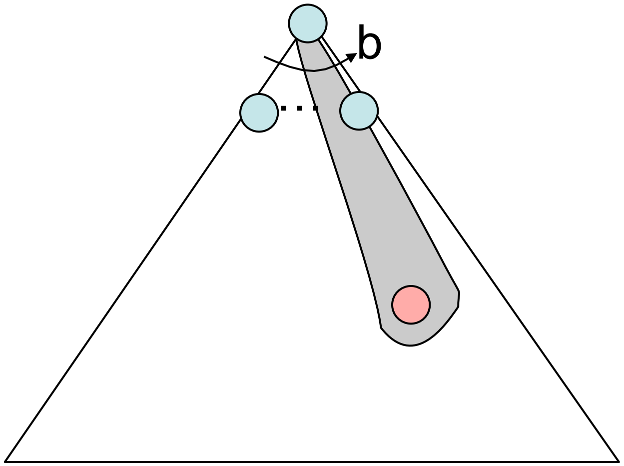
- Expand the node that seems closest…

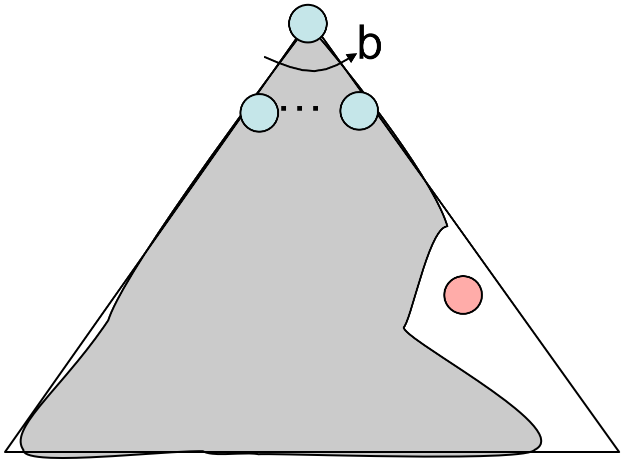


- What can go wrong?

Best First / Greedy Search

- A common case:
  - Best-first takes you straight to the (wrong) goal
- Worst-case: like a badly-guided DFS in the worst case
  - Can explore everything
  - Can get stuck in loops if no cycle checking
- Like DFS in completeness (finite states w/ cycle checking)





To Do:

- Look at the course website:
  - http://www.cs.washington.edu/cse473/14sp
- Do the readings (Ch 3)
- Do PS0 if new to Python
- Start PS1, when it is posted
  - START PS1 ASAP