Announcements

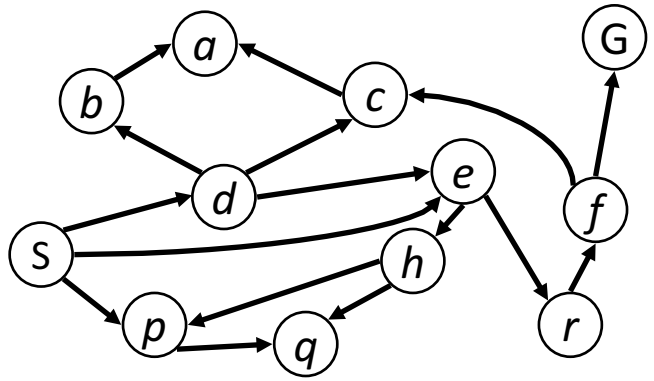
Review DFS, BFS and UCS and **Introduce** heuristic today.

Project 1 out, due next Friday.

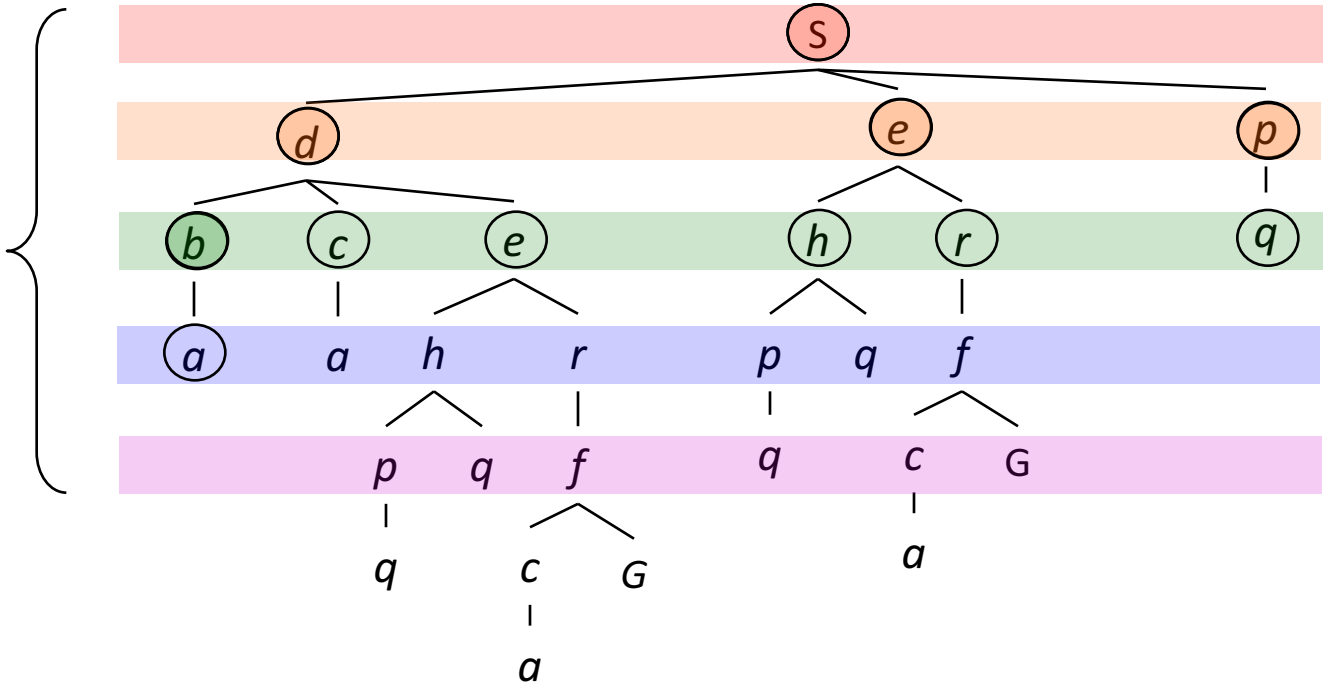
Breadth-First Search

Strategy: expand a shallowest node first

Implementation: Fringe is a FIFO queue



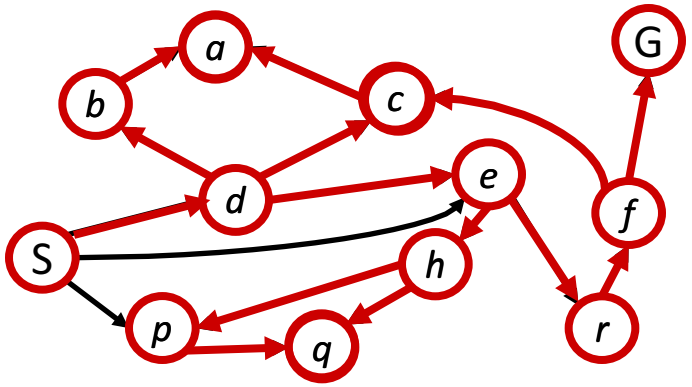
Search Tiers

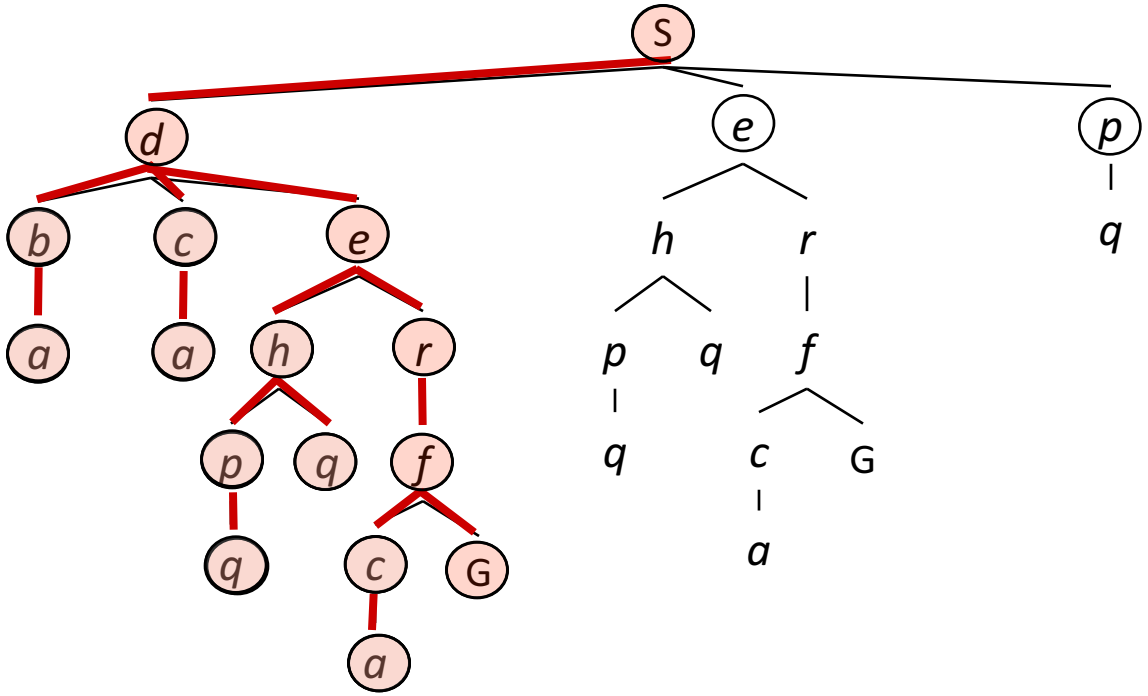


Depth-First Search

Strategy: expand a deepest node first

Implementation: Fringe is a LIFO stack

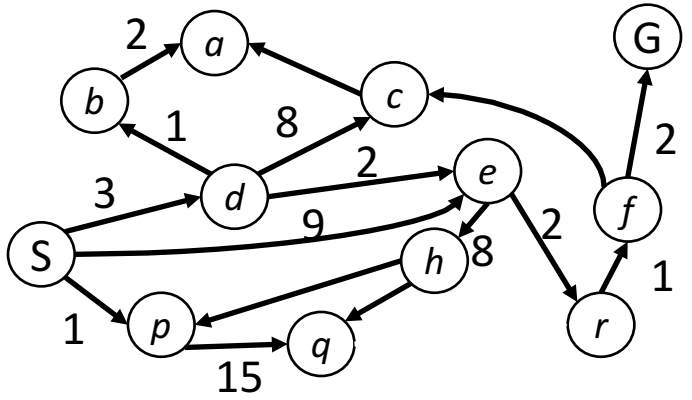




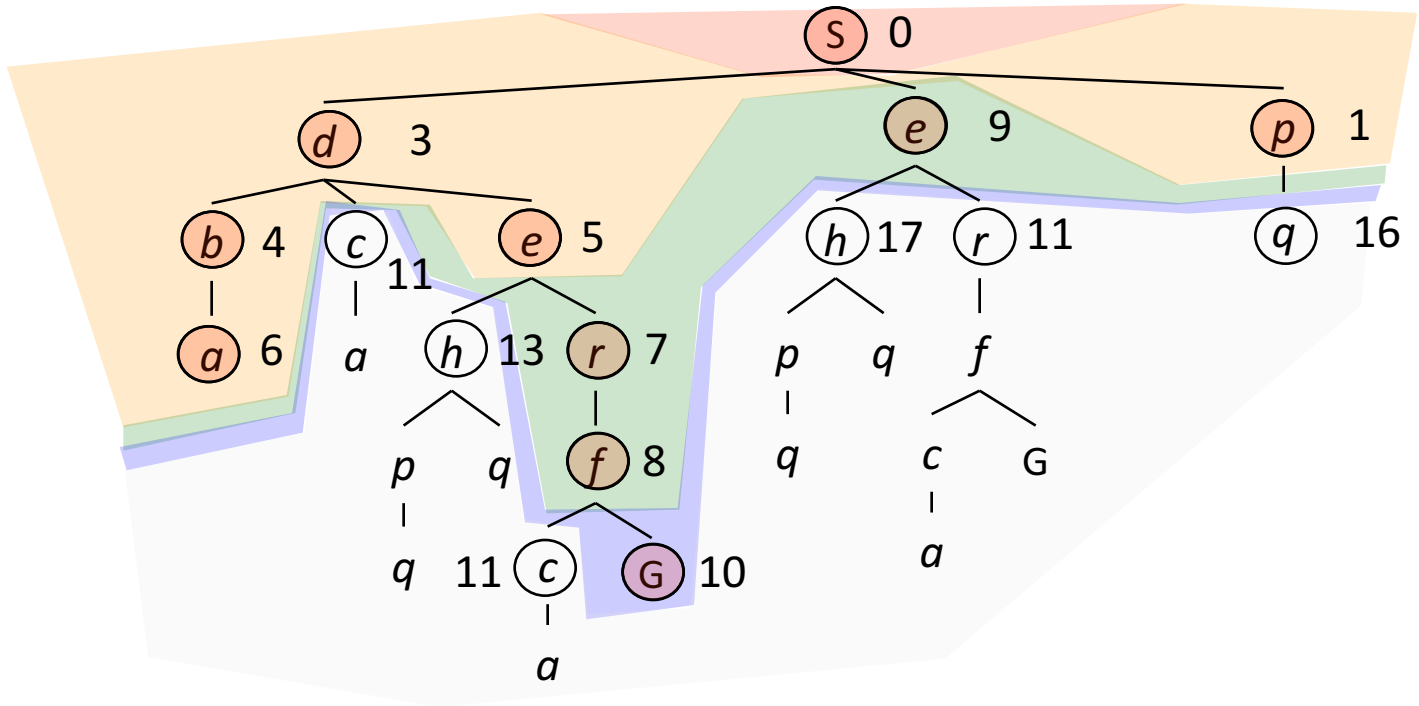
Uniform Cost Search

Strategy: expand a cheapest node first:

Fringe is a priority queue (priority: cumulative cost)



Cost contours



BFS/DFS/UCS

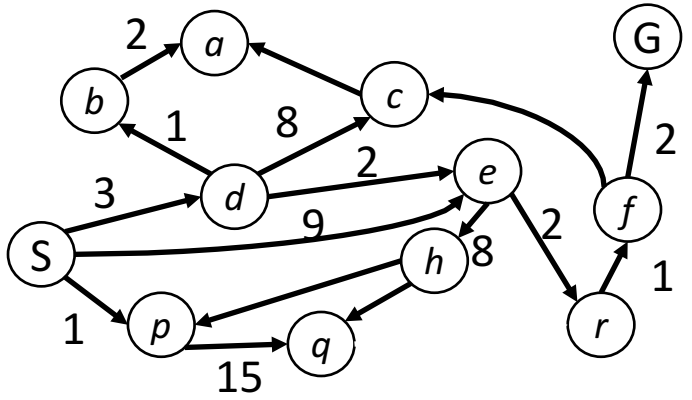
- **Breadth-first search**
  - Good: optimal, works well when many options, but not many actions required
  - Bad: assumes all actions have equal cost
- **Depth-first search**
  - Good: memory-efficient, works well when few options, but lots of actions required
  - Bad: not optimal, can run infinitely, assumes all actions have equal cost
- **Uniform-cost search**
  - Good: optimal, handles variable-cost actions
  - Bad: explores all options, no information about goal location

Basically Dijkstra's Algorithm!

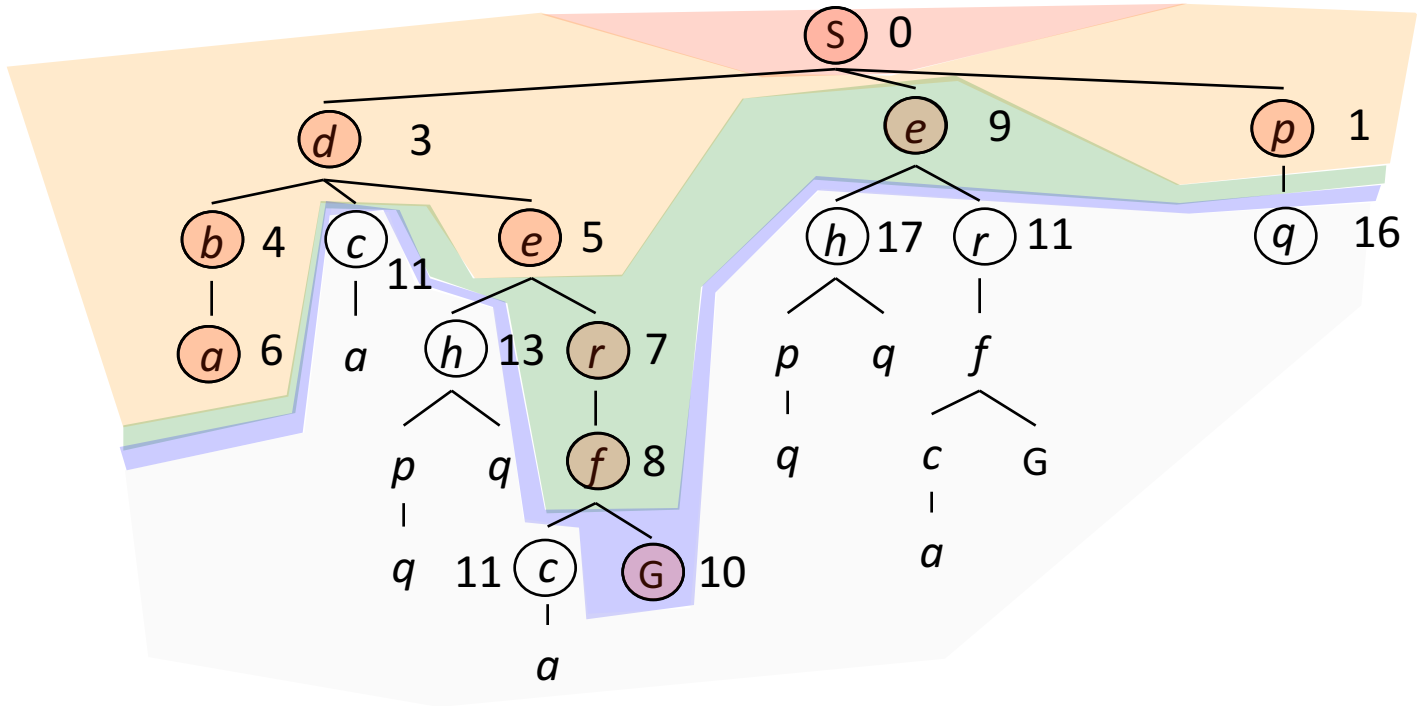
Dijkstra‘s algorithm (Uniform-cost search)

Strategy: expand a cheapest node first:

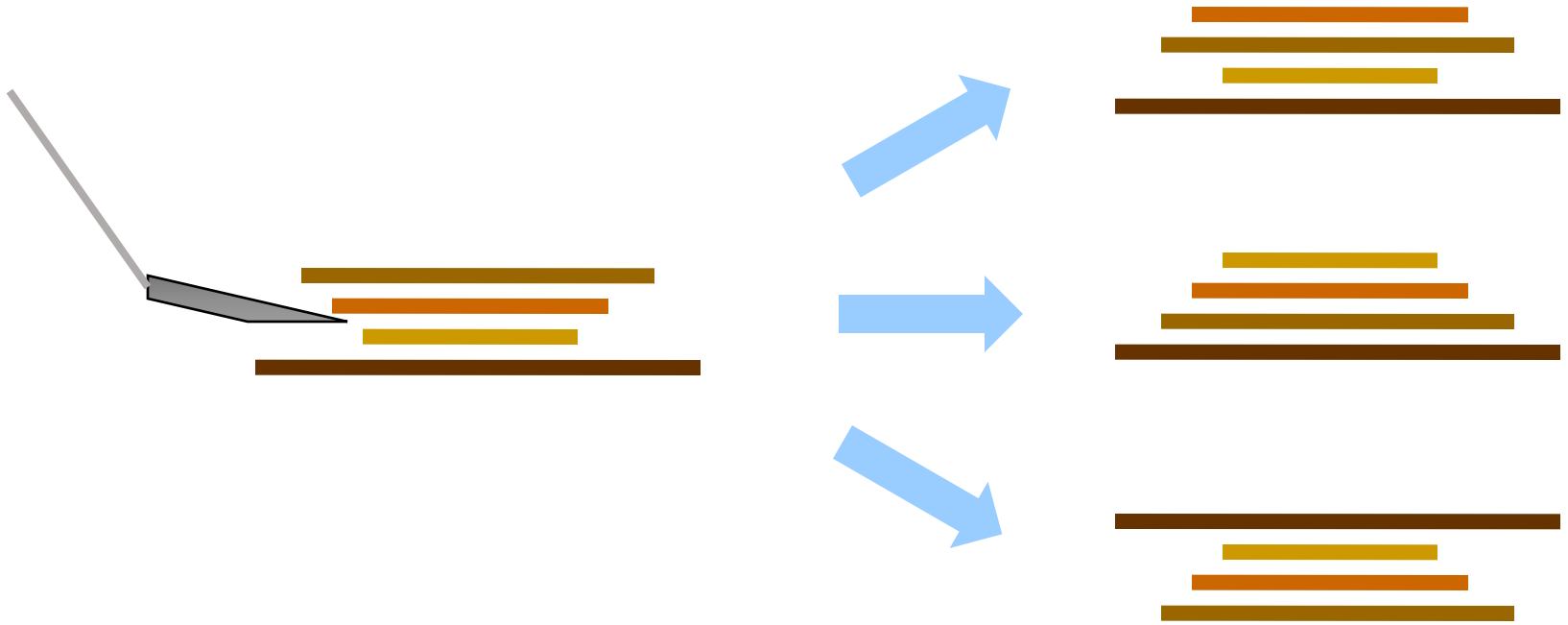
Fringe is a priority queue (priority: cumulative cost)



Cost contours



Search example: Pancake Problem



Rule: a spatula can be inserted at any interval and flip all pancakes above it.

Cost: Number of pancakes flipped.

Pancake BFS

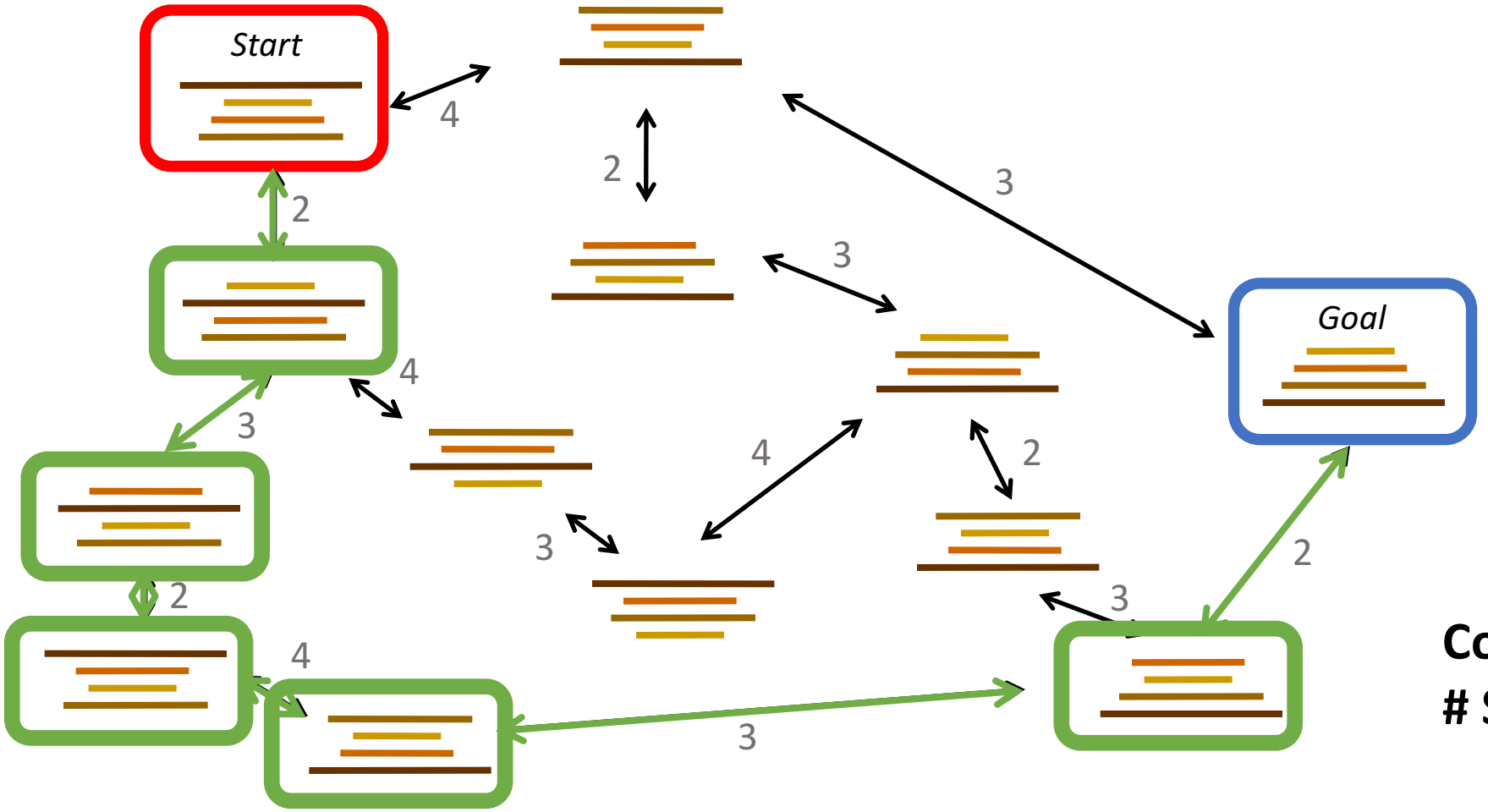
Draw it by yourself!

Pancake UCS

Draw it by yourself!

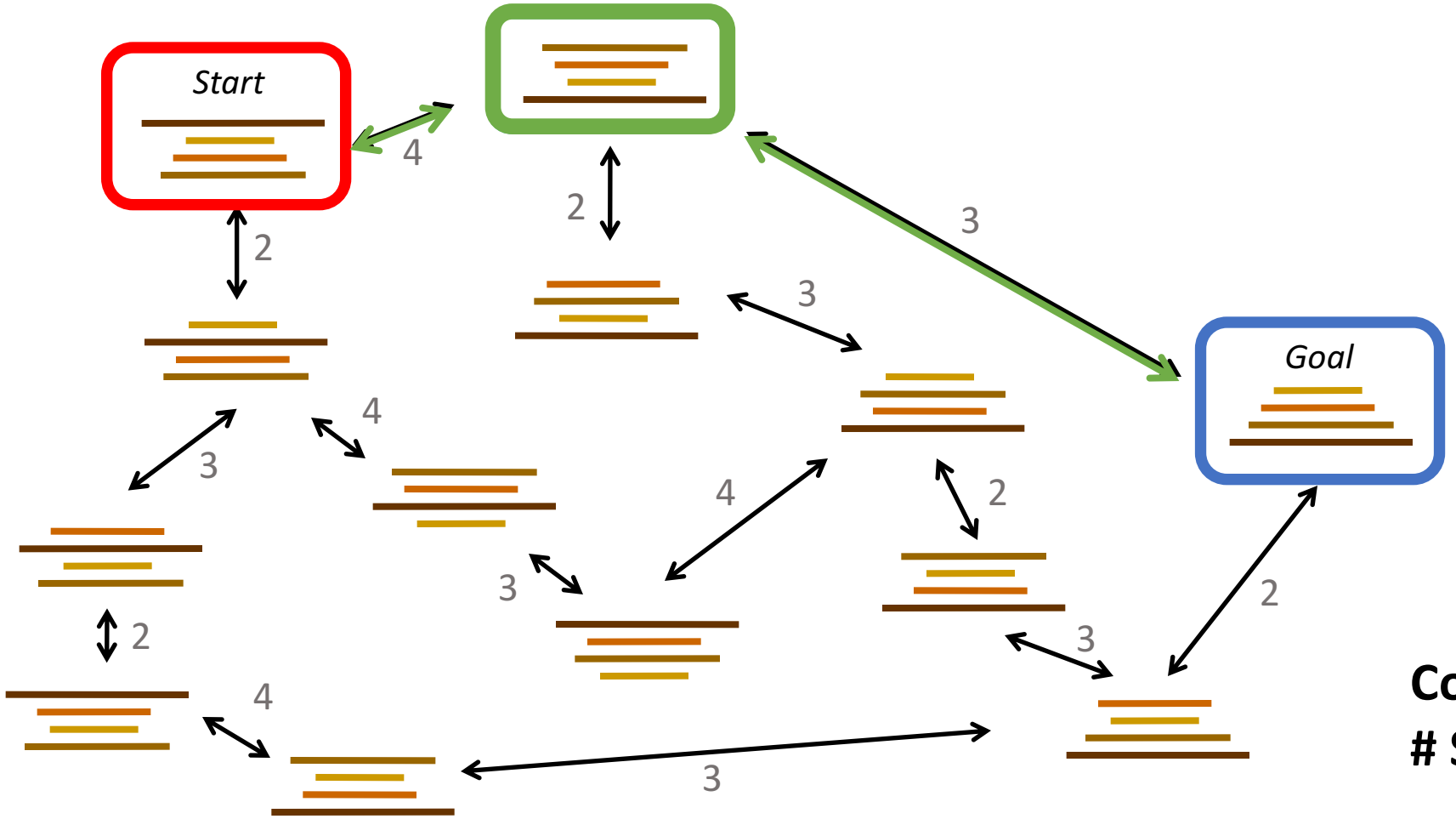
Pancake DFS

State space graph with costs as weights



Pancake Optimal

State space graph with costs as weights



Cost: 7

 # Steps: 2

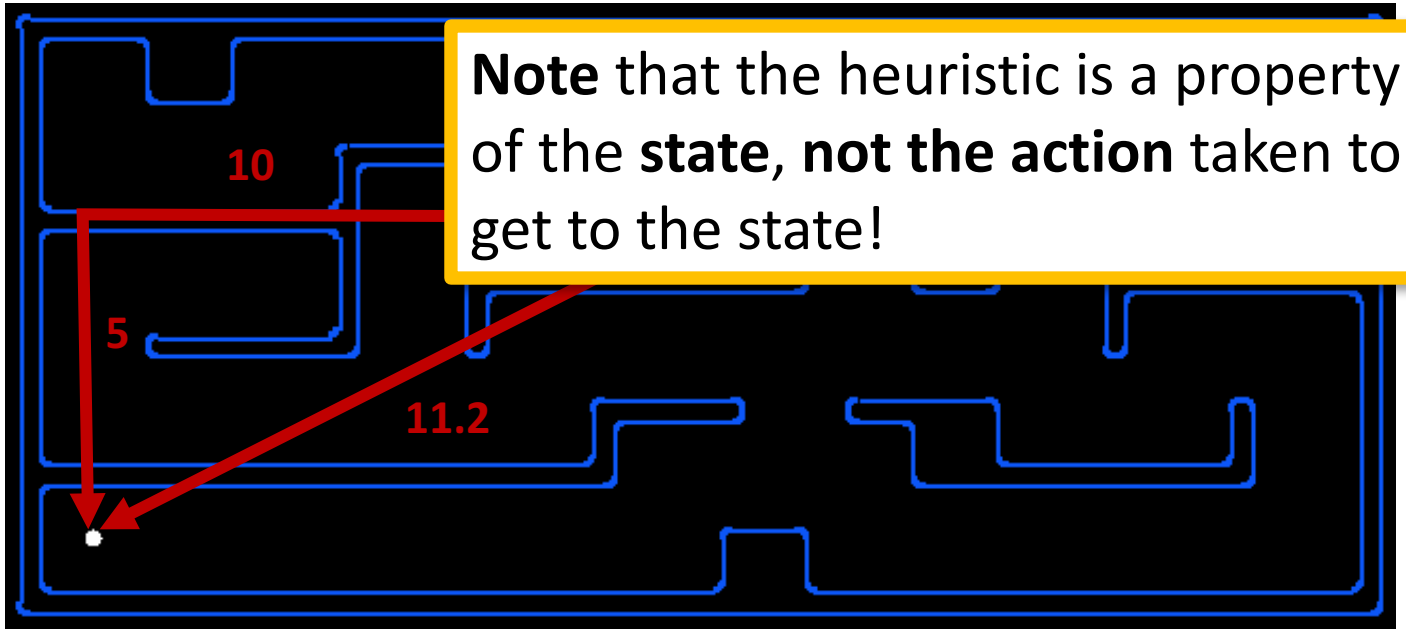
Incorporating goal information

How to efficiently solve search problems with variable-cost actions, using information about the goal state?

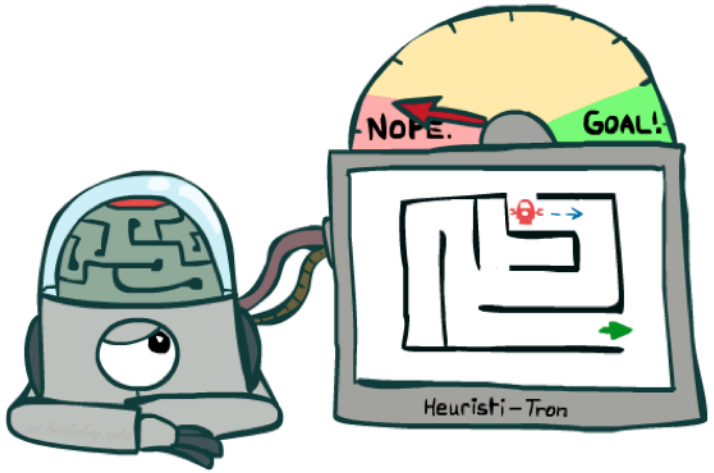
- Heuristics
- Greedy approach
- A* search

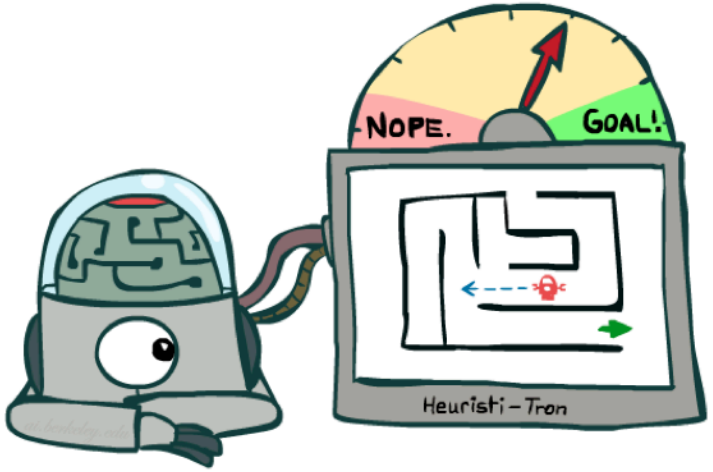
Search Heuristics

- A heuristic is:
  - A function that *estimates* how close a state is to a goal
  - Designed for a particular search problem
  - Examples: Manhattan distance, Euclidean distance for pathing



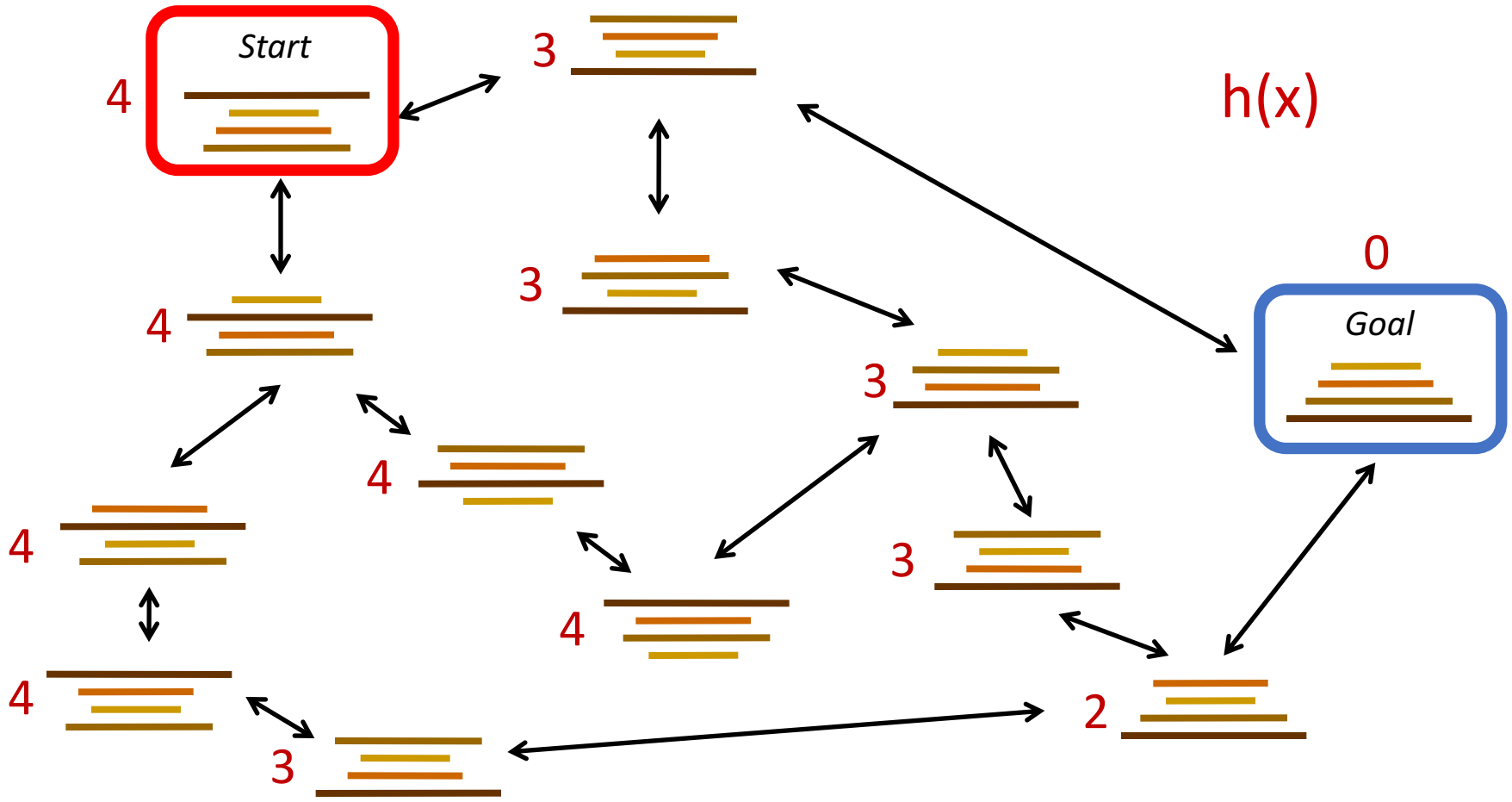
Note that the heuristic is a property of the **state**, **not the action** taken to get to the state!





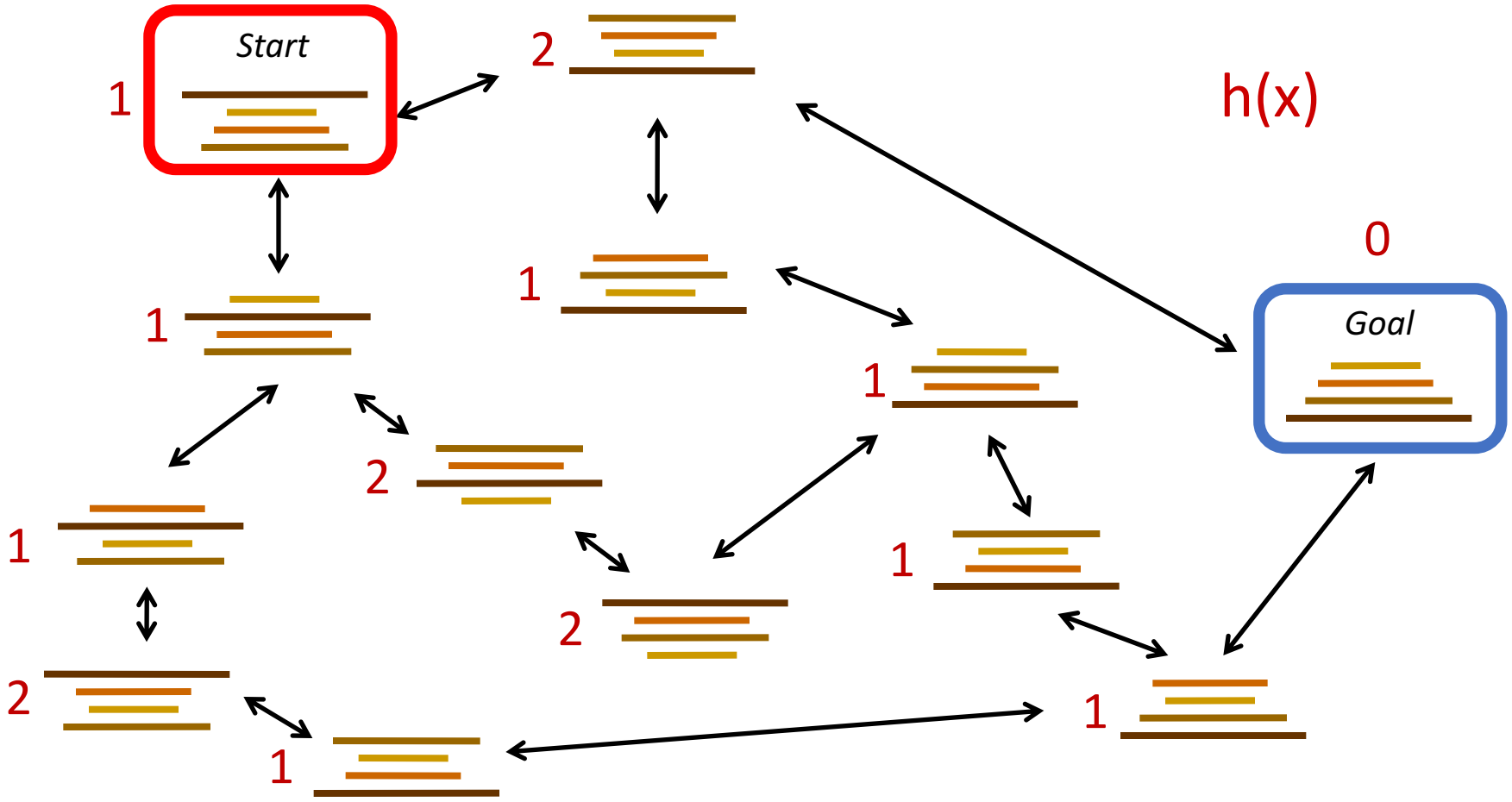
Pancake Heuristics

Heuristic 1: the number of pancakes that are out of place



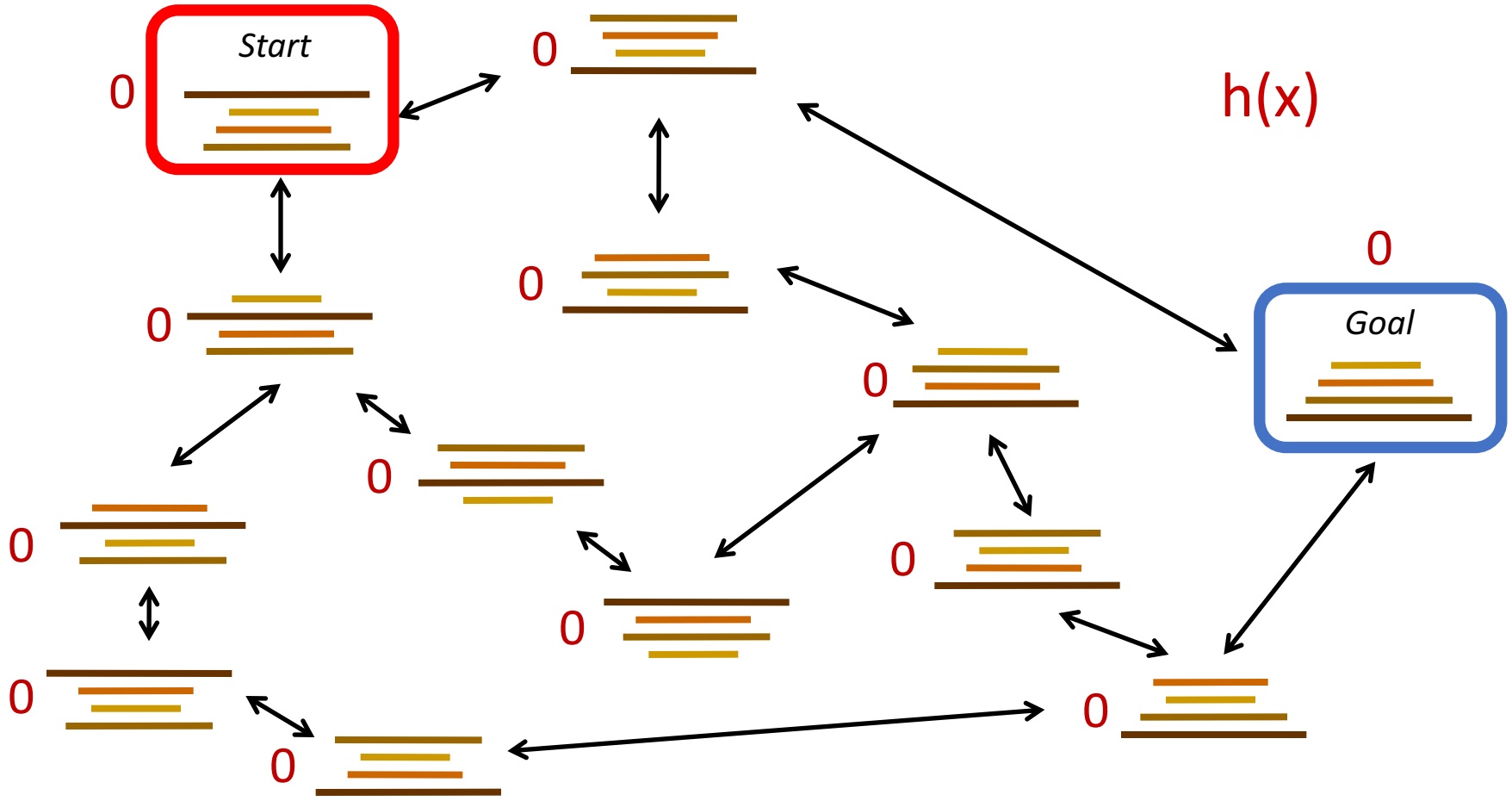
Pancake Heuristics

Heuristic 2: how many pancakes are on top of a smaller pancake?

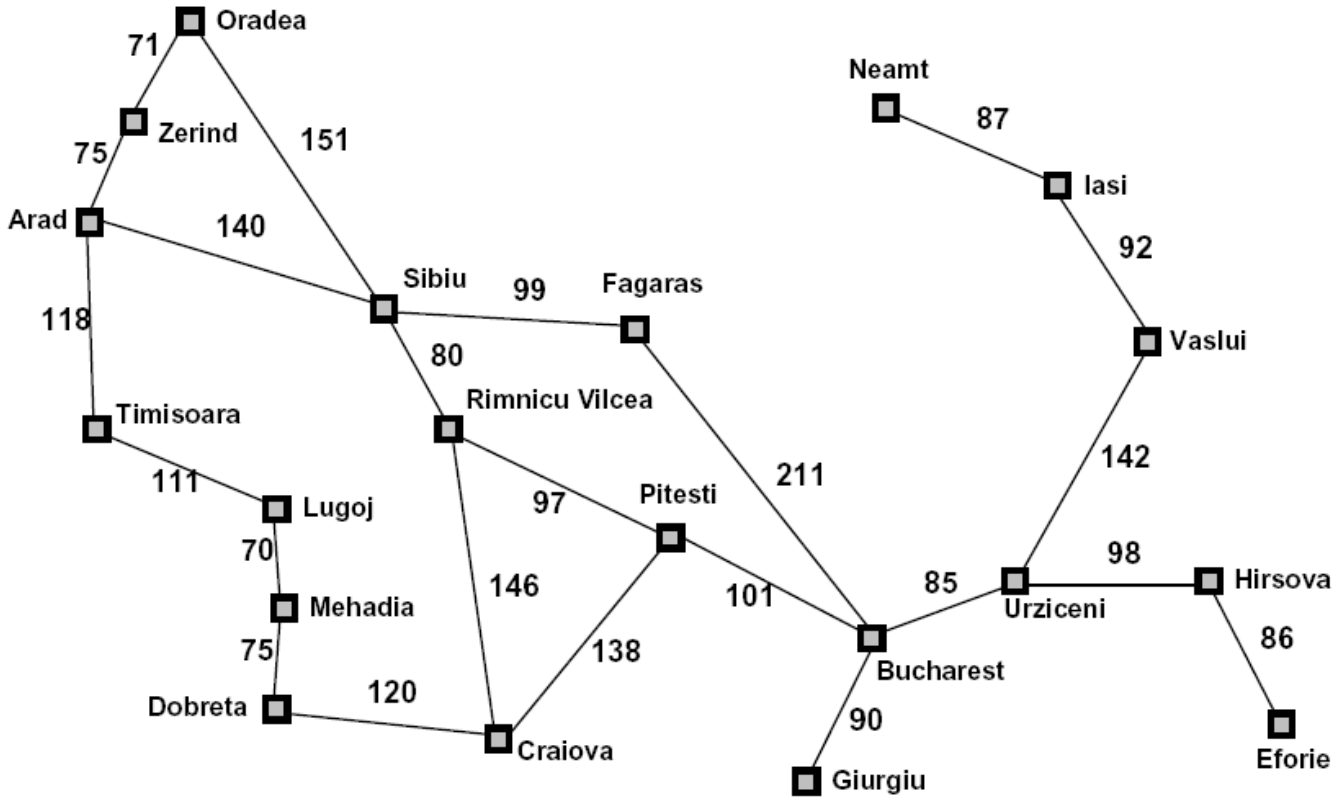


Pancake Heuristics

Heuristic 3: All zeros (aka *null heuristic*, or "I like waffles better anyway")



Straight-line Heuristic in Romania



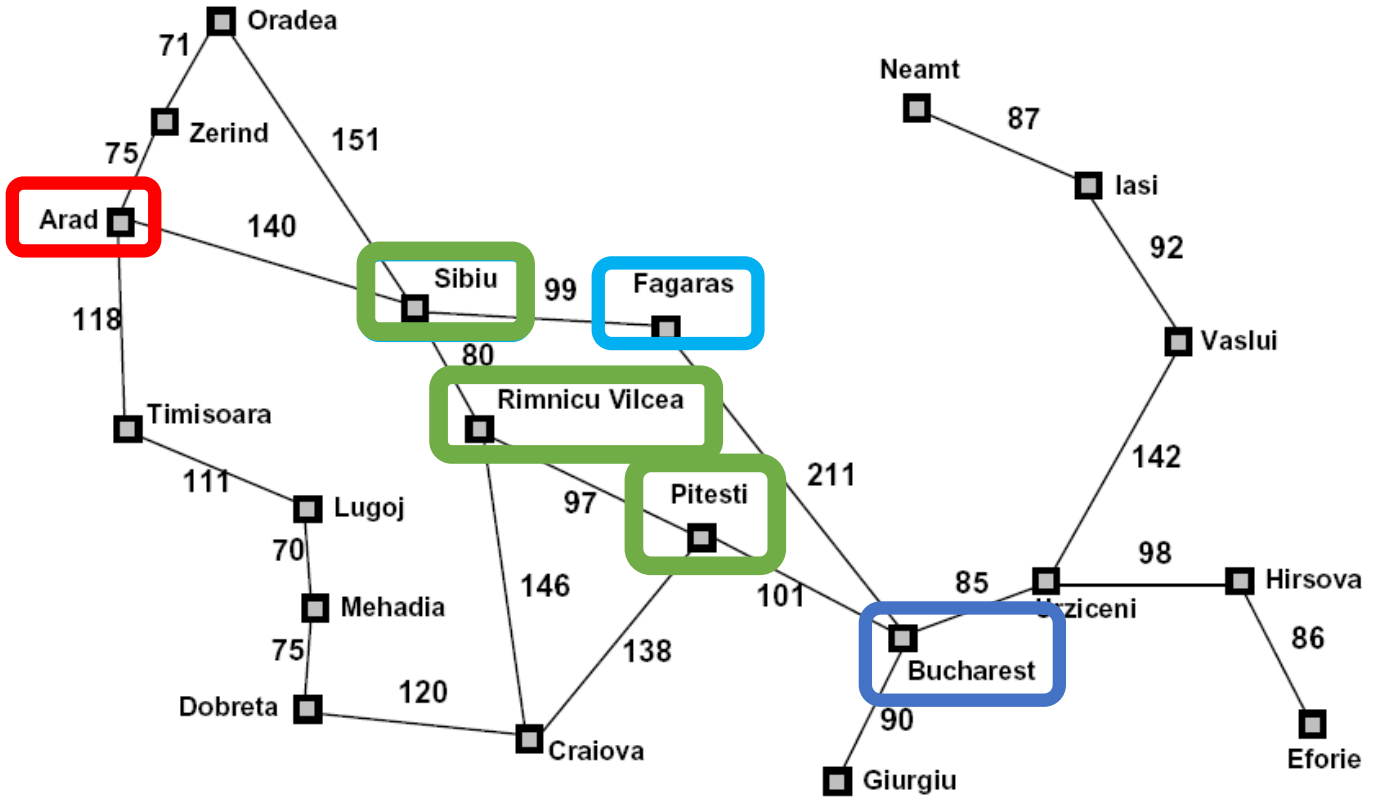
Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

h(x)

Greedy Search

Greedy Straight-Line Search in Romania

- Expand the node that seems closest…



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)

Greedy

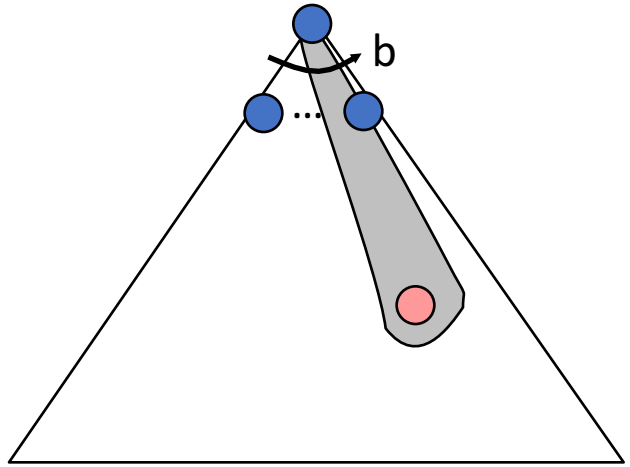
- Cost: 450

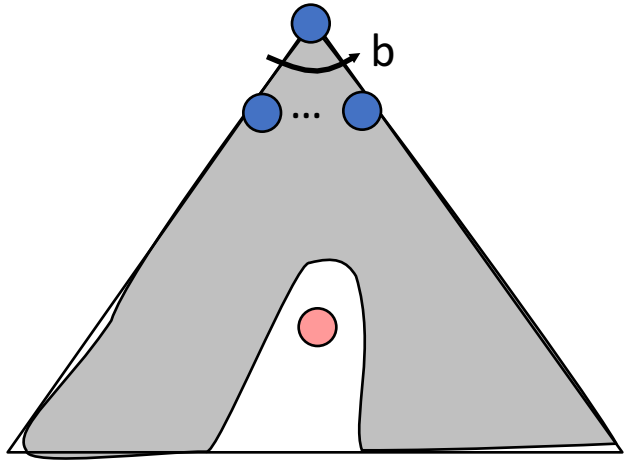
Optimal

- Cost: 418

Greedy Search

- Strategy: expand a node that you think is closest to a goal state
  - Heuristic: estimate of distance to nearest goal for each state
- A common case:
  - Best-first takes you straight to the (non-optimal) goal
- Worst-case: like a badly-guided DFS
- What goes wrong?
  - Doesn't take real path cost into account





Next class

A* search