Basic Search

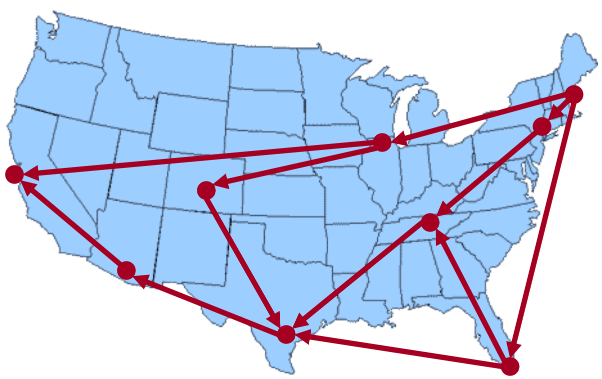
CPSC 470 – Artificial Intelligence

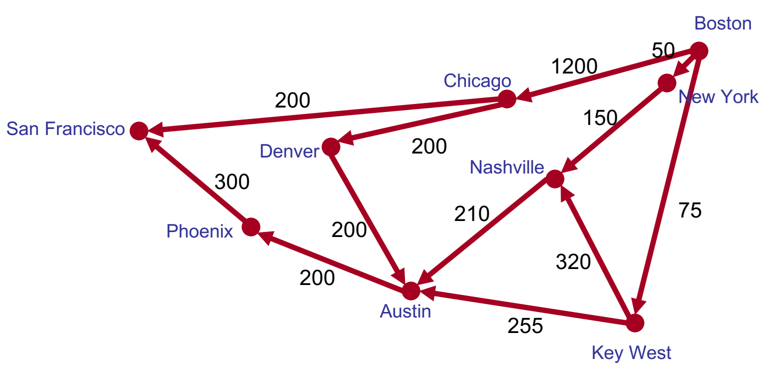
Brian Scassellati

Characterizing Sample Environments

Environment	Observable	Deterministic	Episodic	Static	Discrete
	Do sensors give complete world state?	Can next state be determined by current state and action?	Does quality of an action depend only on current state?	Does the env. stay the same while the agent thinks?	Are the number of percepts and actions limited?
Chess (no clock)	Fully	Yes	No	Yes	Yes
Poker	Partially	No	No	Yes	Yes
Taxi driving	Partially	No	No	No	No
Image analysis	Fully	Yes	Yes	Semi	No
Part-picking robot	Partially	No	Yes	No	No

Problem Formulation



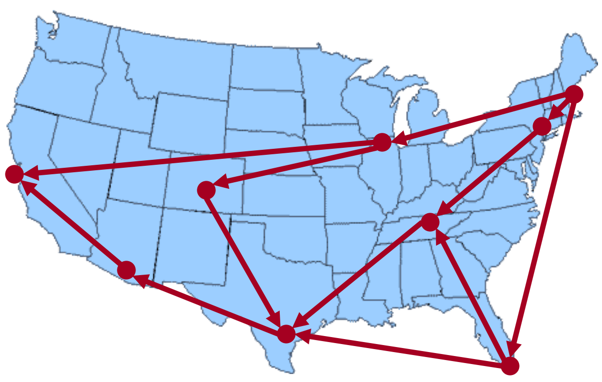


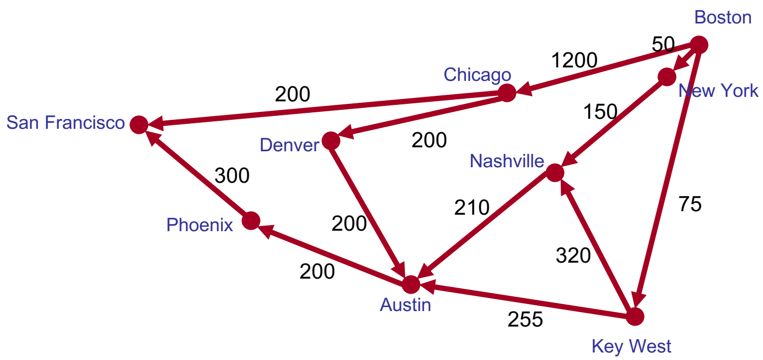
- Well-defined function that identifies both the goal states and the conditions under which to achieve the goal
  - Fly from Boston to San Francisco
  - Quality might depend on
    - Least amount of money
    - Fewest number of transfers
    - Shortest amount of time in the air
    - Shortest amount of time in airports

Problem Formulation

- Well-defined problems
  - Fully observable
  - Deterministic
  - Static
  - Discrete set of possible actions (operations)
- State space: the set of all states that are reachable from an initial state by any sequence of actions
- Path: sequence of actions leading from one state to another

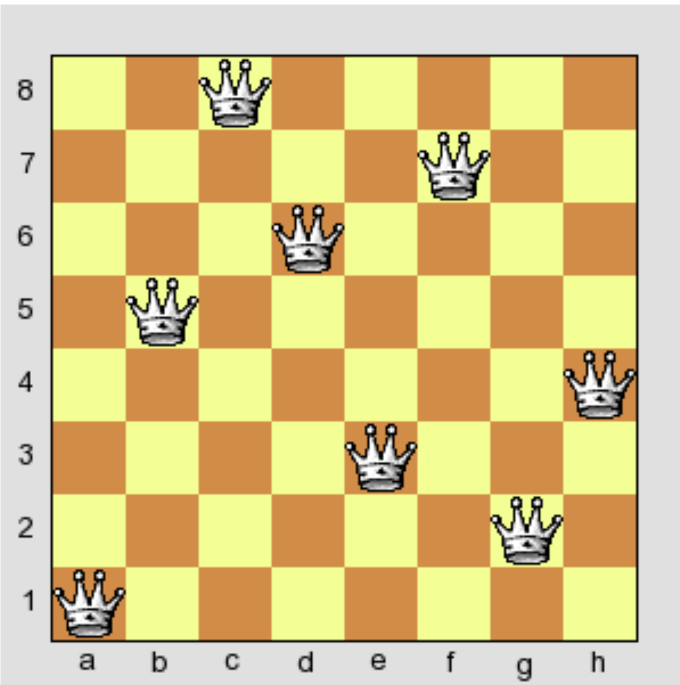
Problem Formulation





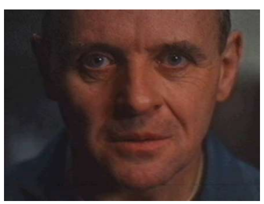
- Goal: spend less $
- State space: flights and their costs
- Path: sequence of flights
- Picking the right level of abstraction
  - Fly from Boston to Chicago
  - Directions to the airport
  - Move left leg 18 inches forward

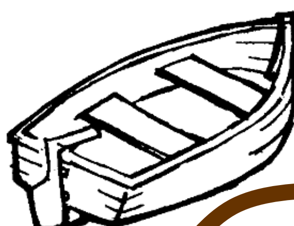
Problem Formulation Matters!
The 8 Queens Problem

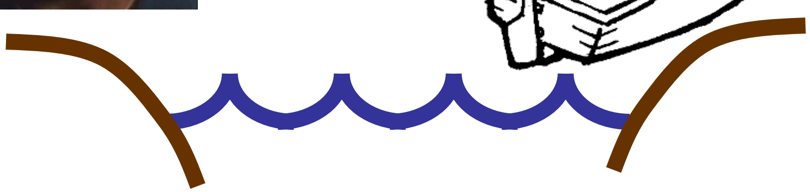


- Formulation #1:
  - Place a queen on any open square
  - Repeat until all queens are placed
  - State space of 64*63*62*61*60*59*58*57=1.78*10^{14}
- Formulation #2:
  - Place a queen on any square in row 1
  - Place a queen on any square in row 2
  - State space of 8*8*8*8*8*8*8*8=1.68*10^{7}
- Formulation #3:
  - Place a queen on any square in row 1
  - Place a queen on a square in row 2 that is not in the same column…
  - State space of 8*7*6*5*4*3*2*1=40,320

Problem Formation involves Abstraction: Missionaries and Cannibals



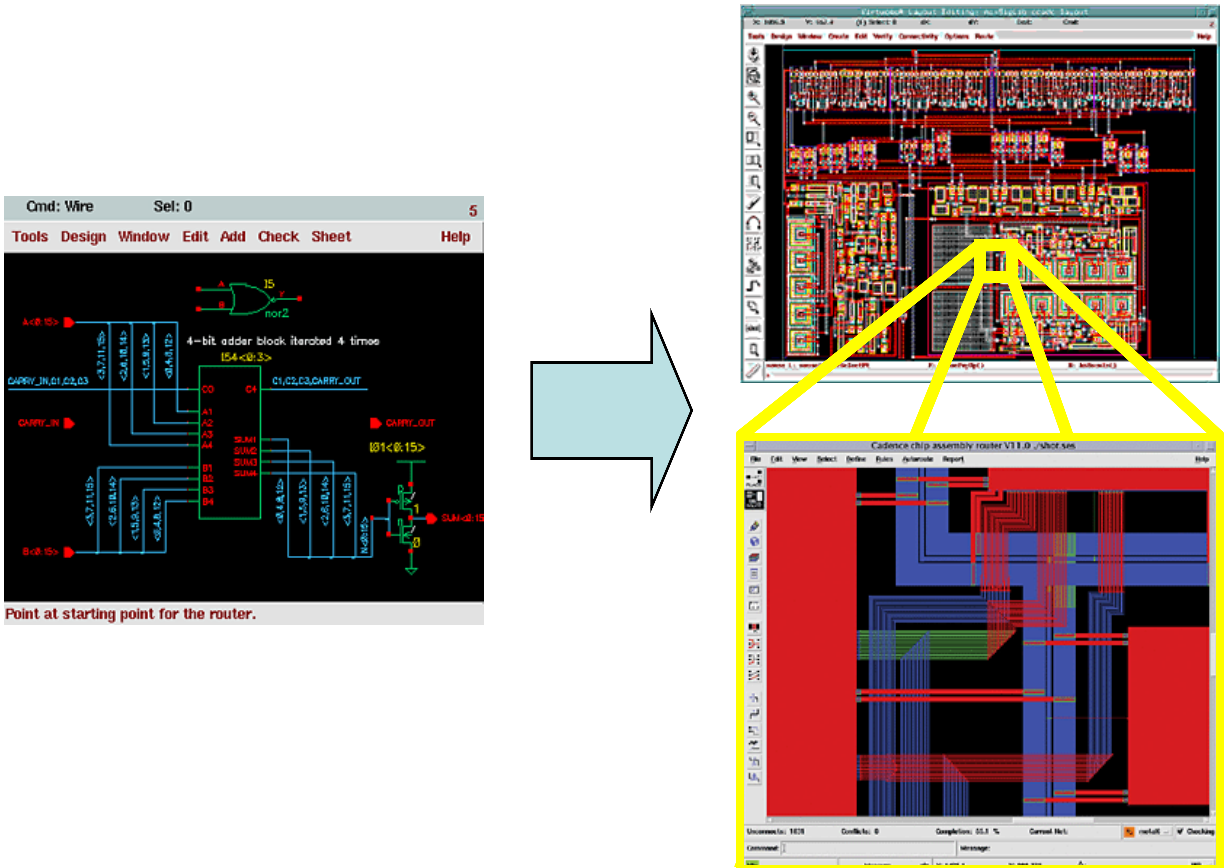




- 3 missionaries and 3 cannibals on left side
- Boat holds 1 or 2 people
- Never leave missionaries outnumbered by cannibals
- States:

  (# cannibals,
   # missionaries,
   # boats) on left side of river
- Operators
  – Remove up to 2 people to other side

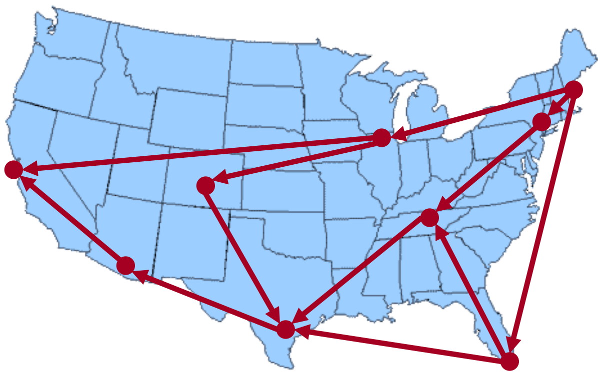
Real-World Applications: VLSI Layout

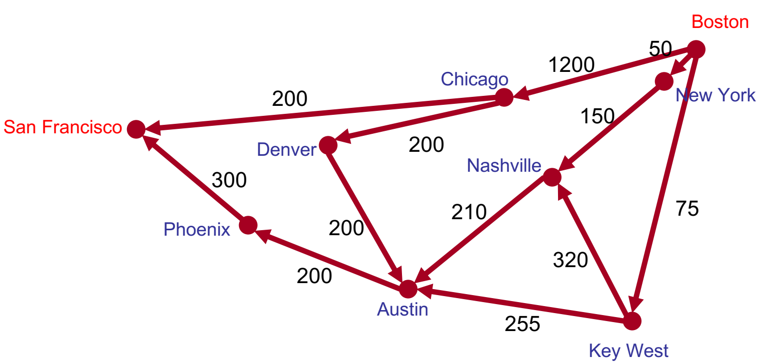


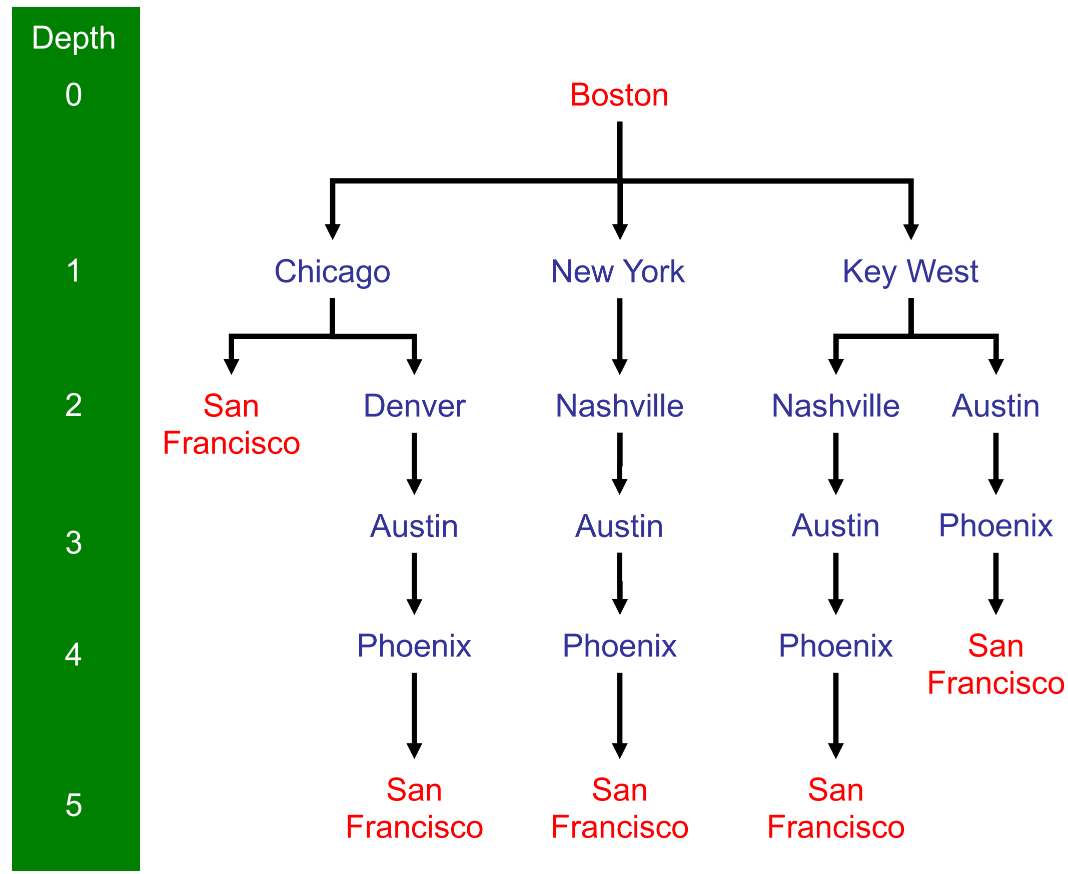
(Images from Cadence Inc.'s Virtuoso System)

Real-World Applications: Traveling Salesman Problem

How to Search:
Generating Sequences and Data Structures





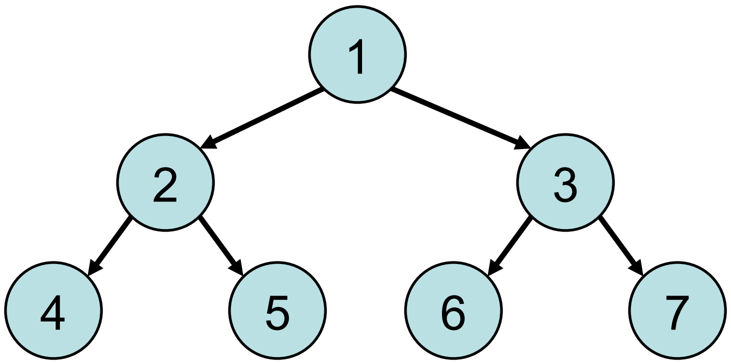


Branching Factor *b*=3

Measuring Performance

- Completeness: is the strategy guaranteed to find a solution when one exists?
- Time Complexity: how long does it take to find a solution?
- Space Complexity: how much memory does it require to perform the search?
- Optimality: Does the strategy find the best-quality solution when more than one solution exists?

Breadth-First Search

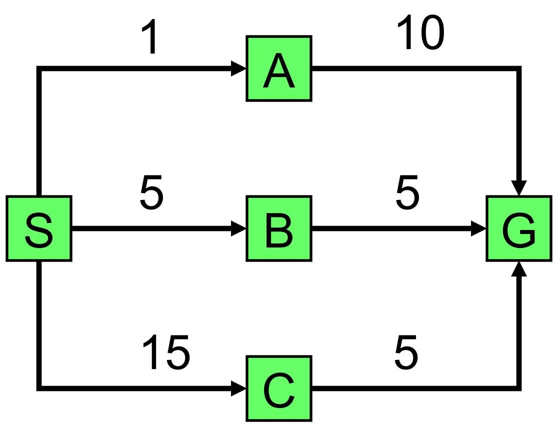


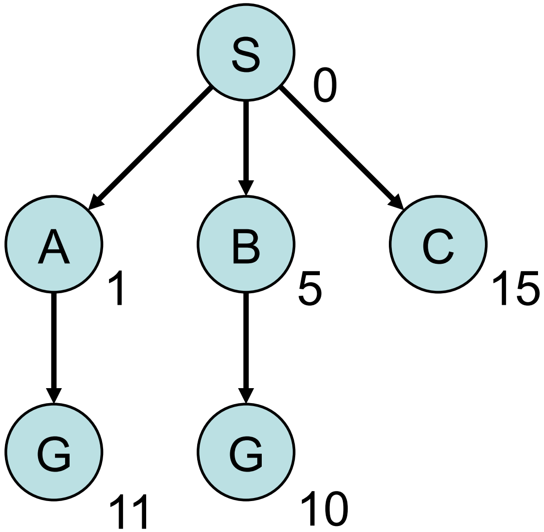
Depth	Nodes	Time	Memory
0	1	1 millisecond	100 bytes
2	111	.1 seconds	11 kilobytes
4	11,111	11 seconds	1 megabyte
6	10^6	18 minutes	111 megabytes
8	10^8	31 hours	11 gigabytes
10	10^{10}	128 days	1 terabyte
12	10^{12}	35 years	111 terabytes
14	10^{14}	3500 years	11,111 terabytes

- Finds the most shallow solution
- Complete
- Optimal when the path cost is a non-decreasing function of depth

- Assuming
  - Branching factor b=10
  - Process 1000 nodes/sec
  - 100 bytes/node
- Time is a big issue
- Space is a bigger issue
- Exponential growth leads to impractical problems for uninformed search

Uniform Cost Search

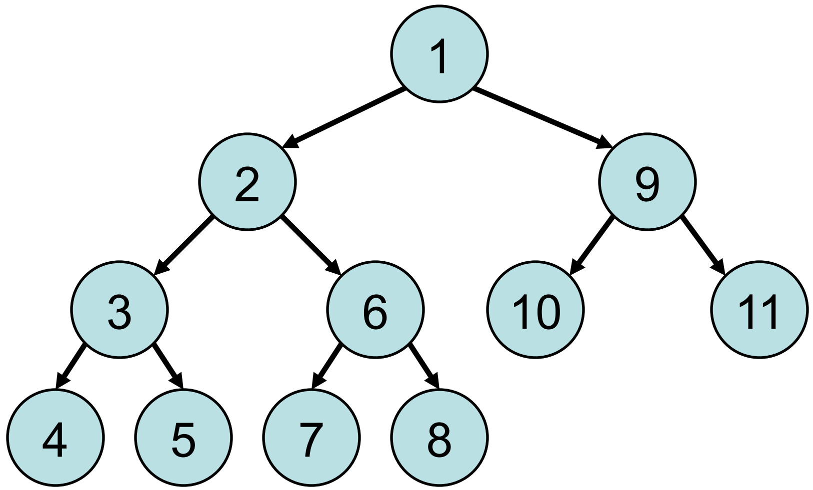




- Travel from the start (S) to the goal (G)
- Cost associated with each link

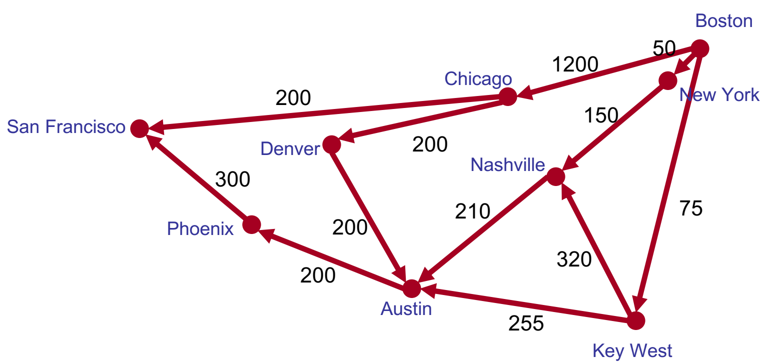
- Always expand the fringe node with the lowest cost
- Breadth-first search is uniform search with cost=depth

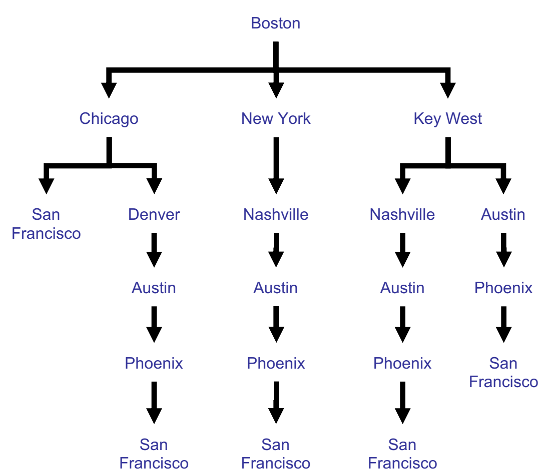
Depth-First Search



- Minimal memory requirements (only stores one path at a time)
- Best case scenario
- Worst case scenario
- Non-optimal
- What happens on trees with infinite depth?
  - Completeness is sacrificed

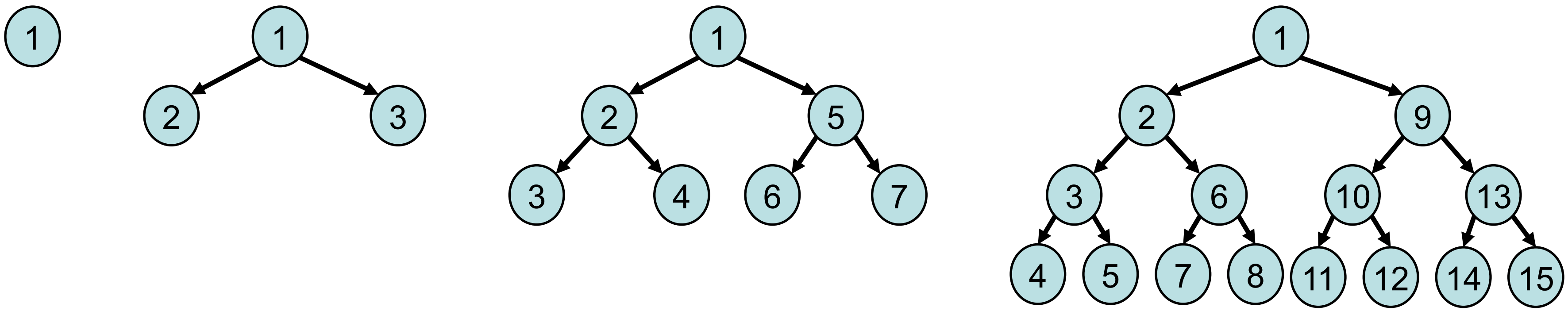
Depth Limited





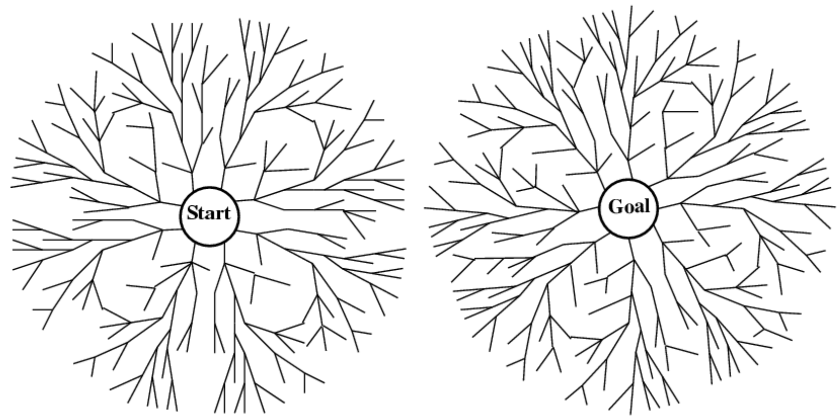
- Follow depth-first search, but with a maximum depth
- Requires some knowledge of the solution:
  - 9 cities, depth limit of 8?
- Non-optimal
- What if we choose a limit too small?
  - Sacrifice completeness

Iterative Deepening



- Tries all possible depth limits (l=0,1,2,…)
- Cost of re-computing the lower depths
  - But most nodes are in the deep bottom of the tree
  - Tree with depth 3, branching factor 2
    - 1+2+4+8 = 15 nodes for pure depth first search
    - 3+7+15 = 25 nodes for iterative deepening search
  - Tree with depth 5, branching factor 10
    - 1+10+100+1,000+10,000+100,000 = 111,111 nodes depth-first
    - 6+50+400+3,000+20,000+100,000 = 123,456 nodes iterative depth

Bidirectional Search



- If you can work backward from the solution, then you can limit the search depth
- With a solution at depth *d*, then find a solution in $O(2b^{d/2})=O(b^{d/2})$ steps
- Better than $O(b^d)$ steps with breadth-first search!
  - For a tree with b=10, d=6
    - Breadth-first search generates 1,111,111 nodes
    - Bi-directional search generates 2,222 nodes

Comparison of Techniques

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Time	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Space	b^d	b^d	bm	bl	bd	$b^{d/2}$
Optimal?	Yes	Yes	No	No	Yes	Yes
Complete?	Yes	Yes	No	Yes, if $l \geq d$	Yes	Yes

- b = branching factor
- d = depth of solution
- m = maximum depth of tree
- l = depth limit

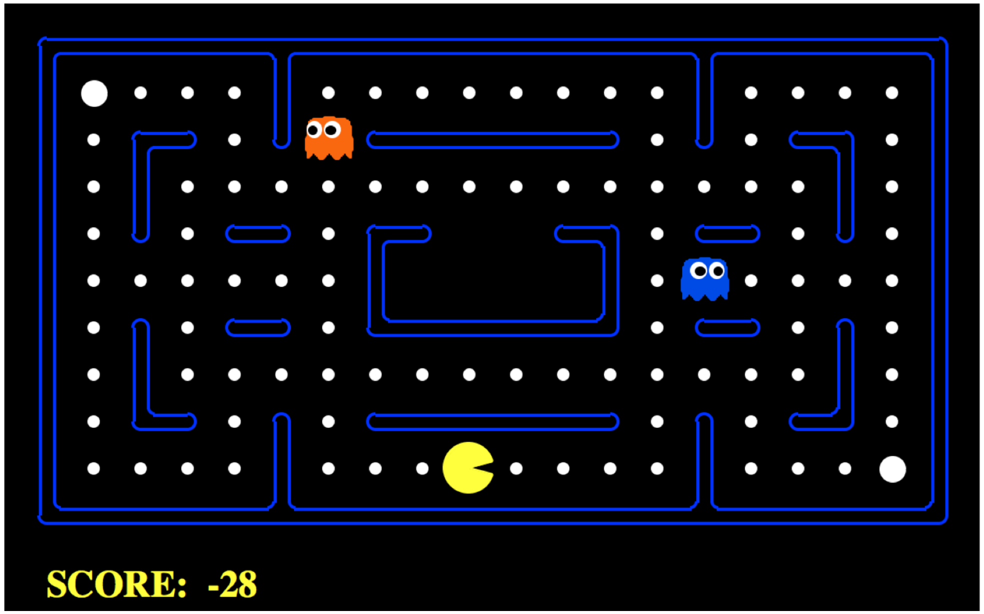
Still not as smart
as it could be…

Coming Up Next

- More intelligent search strategies
  - Best-first search
  - A* search
  - Heuristic search
- Applications
  - Playing games
  - Constraint satisfaction problems

Administrivia

- Office hours posted today
- PS 0 due today at 11:59pm
- PS 1 out today… search in PACMAN



Sign Up to Work on a Collaborative Task with a Robot

HSC#: 2000023736

Location: AKW 500
(51 Prospect Street)

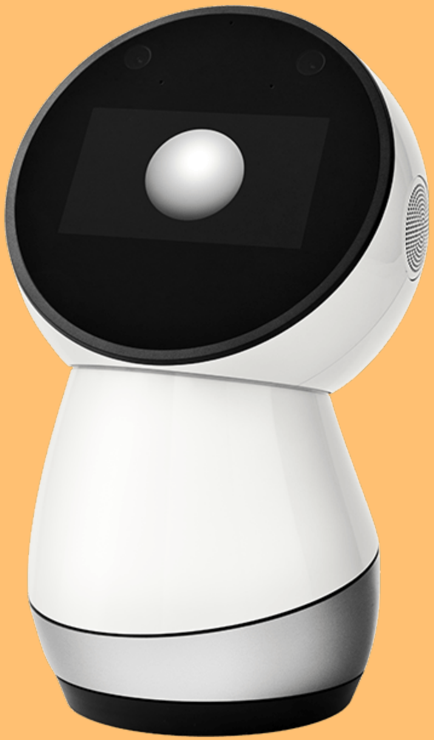
Time commitment:
60 minutes

Please contact
sarah.sebo@yale.edu
with any questions

And earn

$10!

Sign up at:
tinyurl.com/yale-robot-team-task



tinyurl.com/
yale-robot-
team-task