****
CS512 -- Lecture Week 2,3
****

Agents that Plan/Search
(Chapter 7)
++++++++++

Problem:

(1) start state

(2) solution, goal state

Flavor 1: Problem Solving

- Given start state,
  find the goal state
  (or, solution)

Flavor 2: Planning

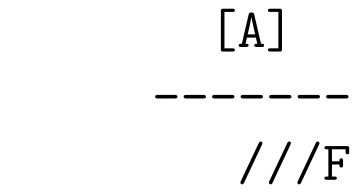
- Given start state, find the
  sequence of moves that will
  produce the goal state

State-Space Graphs

- map out all possible results of actions applied to all possible states

- states = snapshots of a ‘‘world’’

  e.g., ‘‘Blocks World’’



Moves:

block X on block Y --- move(X, Y)

block X on Floor --- move(X,Floor)

Possible Moves and States: pp. 119

Finding solutions means to be searching the state-space for a sequence of actions that leads from the start node to the goal node.

Sequence of actions = ‘‘Plan’’

State Space is Typically a GRAPH (directed)

It may be a specialized graph called TREE (directed)

Graph Notation:

- node, arc
- parent, successor
- ancestor, descendent

Tree Notation:

branching factor
(no. arcs out of node)

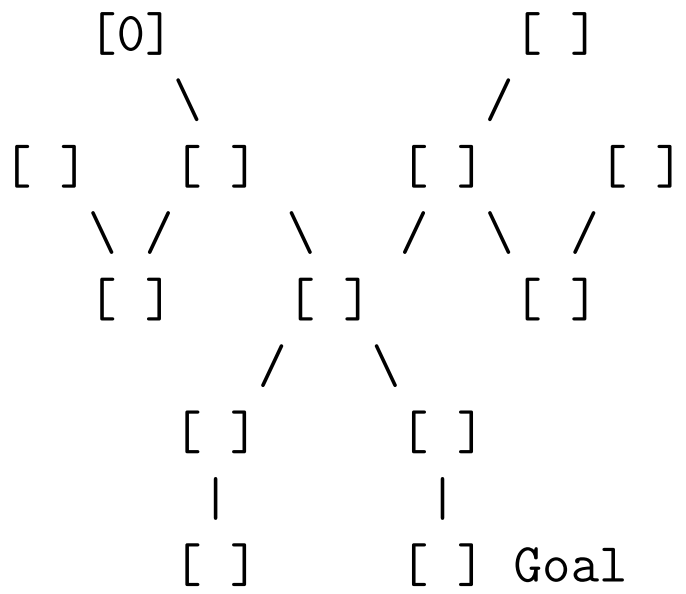
+ root
+ leaf

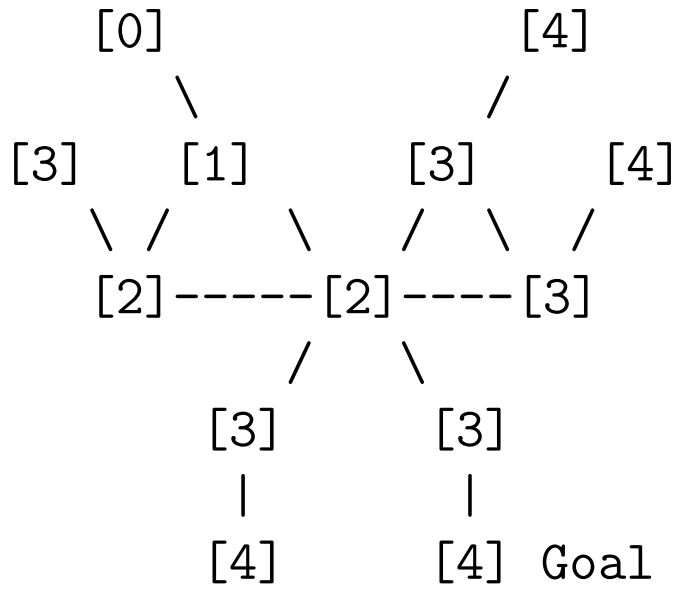
Algorithms to Systematically Search Graphs

(and trees)

- Breadth-First Search
- Depth-First Search
- Others ...

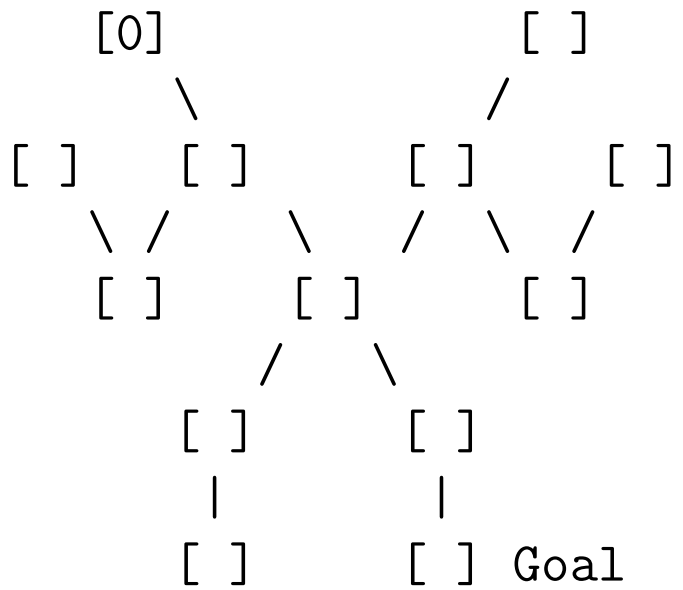
Breadth-First Search:





[i] = node at ith level of EXPANSION

Depth-First Search?



Popular Puzzles (and AI Toy Domains)

- Missionary and Cannibals
- Tower of Hanoi
- Eight Puzzle (***)

All are path-finding problems.

All can be solved by searching the state-space (graph).

--> Uninformed Search

--> Informed/Heuristic Search

EXPANDING a Node

Two possibilities:

(A) Moving from a node to one of its successors in the EXPLICIT state space graph

e.g., in Blocks World, pp. 119

(B) Applying a SUCCESSOR FCT to the node which produces a successor node.

The state space graph is only IMPLICIT (because it may be large to be explicitly listed)

A General Graph-Searching Algorithm
(-> Chapter 9)
++++++++++++++++++++++++++++++++++

GRAPHSEARCH:

1. Create a search tree (Tr) consisting
   of start node n0 only.
   Put n0 on list OPEN.

2. Create a list CLOSED; initially empty.

3. If OPEN is empty, exit with failure.

4. Select first node on OPEN. Remove
   it from OPEN, and put it on CLOSED.
   Call this node n.

5. If n is a goal node, exit successfully.

6. Expand node n, generating a set M of
   successors.

   Add each successor s to OPEN (front,
   or back?) if is not already in

OPEN or CLOSED.

7. Reorder list OPEN according to some criterion. (Or, not.)

8. Go to step 3.

Notice the options we have:

- add node to FRONT of OPEN
- add node to BACK of OPEN

and

- REORDER OPEN
- do not reorder OPEN

Choices give rise to three different search behaviors:

- Breadth-First (back, no reorder)

- Depth-First (front, no reorder)

- Heuristic Search (front/back, reorder)
  (Informed Search)