CS 4260 and CS 5260
Vanderbilt University

Lecture on Uniformed Search

This lecture assumes that you have
- Read Chapter 2 through section 2.3 of ArtInt (15 pages)
- Read Chapter 3 through section 3.5.2 of ArtInt (20 pages) and
- Watched Doug's iterative deepening search video playlist (material from section 3.5.3 optional reading)

ArtInt: Poole and Mackworth, Artificial Intelligence 2E
at http://artint.info/2e/html/ArtInt2e.html
and some slides at https://artint.info/2e/slides/index.html

Iterative Deepening playlist:
https://www.youtube.com/watch?v=7QcoJjSVT38&list=PLXAjOiPf89kPs82cbS6j9PR3t7ZzixnaO

Learning Objectives

At the end of the class you should be able to:

- define a directed graph
- represent a problem as a state-space graph
- explain how a generic searching algorithm works

Searching

- Often we are not given an algorithm to solve a problem, but only a specification of what is a solution — we have to search for a solution.
- A typical problem is when the agent is in one state, it has a set of deterministic actions it can carry out, and wants to get to a goal state.
- Many AI problems can be abstracted into the problem of finding a path in a directed graph.
- Often there is more than one way to represent a problem as a graph.

State-space Search

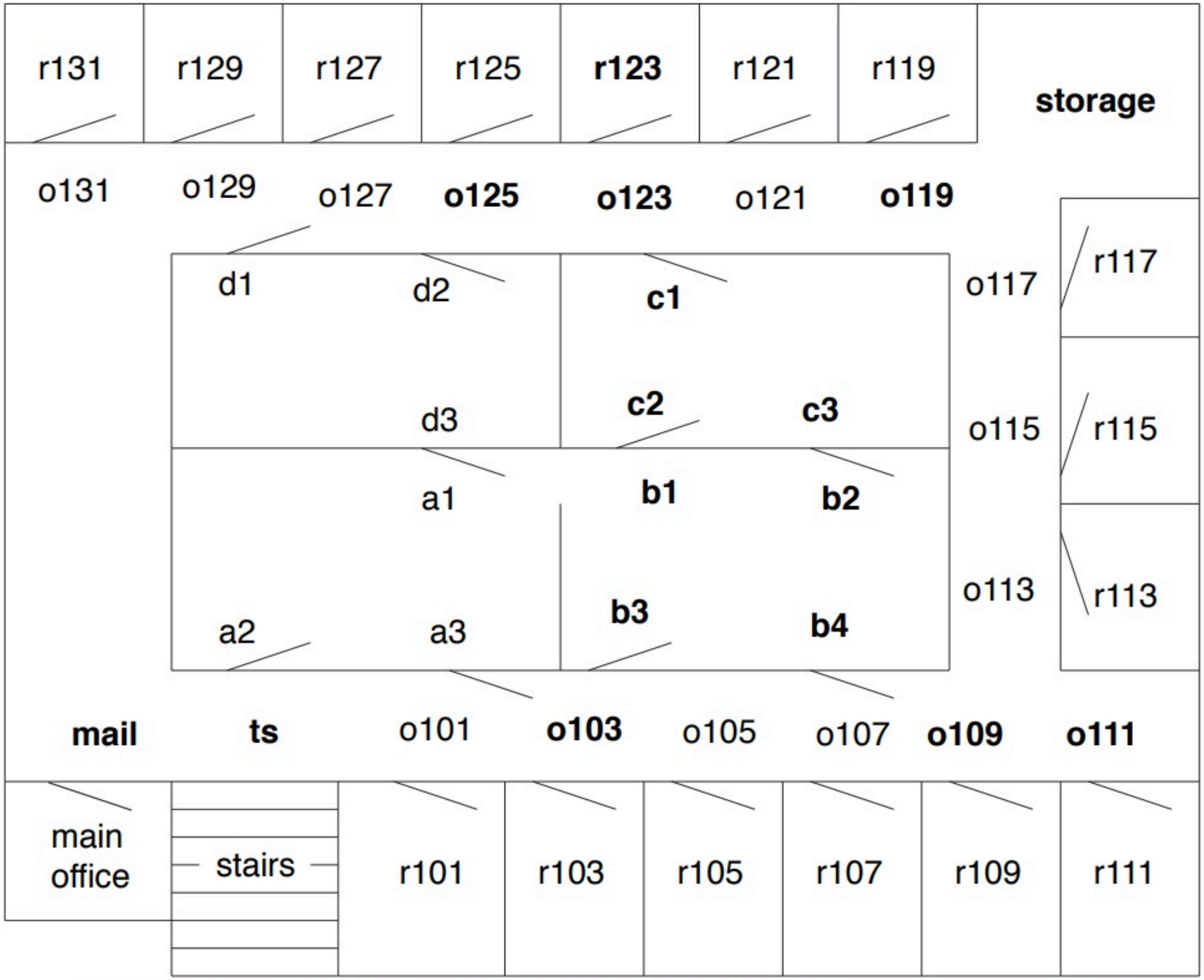
- flat or modular or hierarchical
- explicit states or features or individuals and relations
- static or finite stage or indefinite stage or infinite stage
- fully observable or partially observable
- deterministic or stochastic dynamics
- goals or complex preferences
- single agent or multiple agents
- knowledge is given or knowledge is learned
- perfect rationality or bounded rationality

Directed Graphs

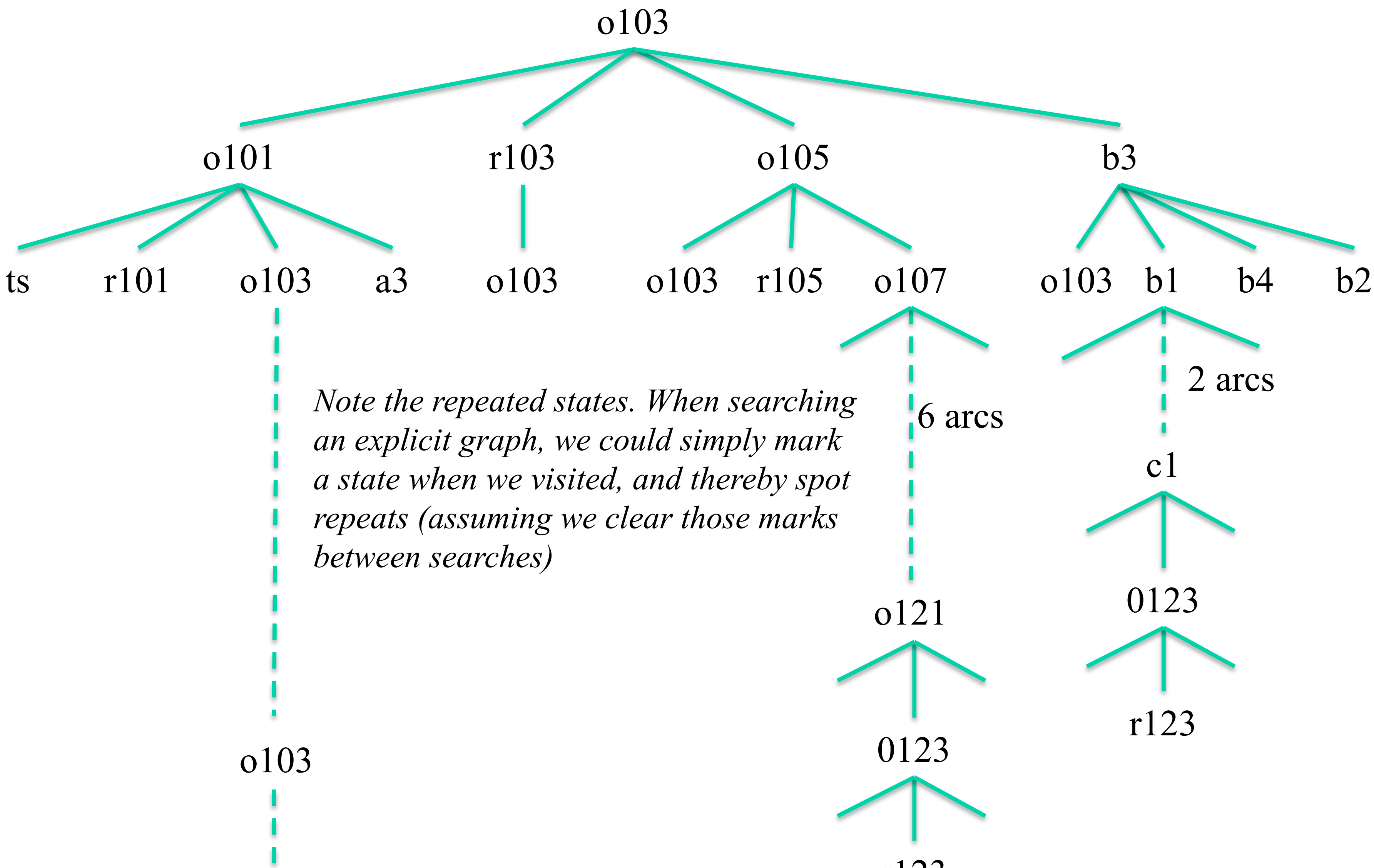
- A (directed) graph consists of a set N of nodes and a set A of ordered pairs of nodes, called arcs.
- Node n_2 is a neighbor of n_1 if there is an arc from n_1 to n_2. That is, if $\langle n_1, n_2 \rangle \in A$.
- A path is a sequence of nodes $\langle n_0, n_1, \ldots, n_k \rangle$ such that $\langle n_{i-1}, n_i \rangle \in A$.
- The length of path $\langle n_0, n_1, \ldots, n_k \rangle$ is k.
- Given a set of start nodes and goal nodes, a solution is a path from a start node to a goal node.

Example Problem for Delivery Robot

The robot wants to get from outside room 103 to the inside of room 123.



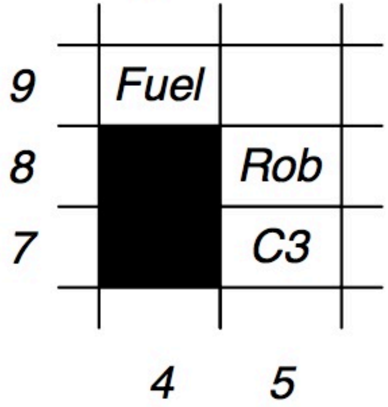
Search tree for Figure 3.1 of Poole and Mackworth, assuming robot starts at 0103, with a goal of being in r123, with all arcs bidirectional (e.g., doors open in both directions, which is a different assumption than text's)



Note the repeated states. When searching an explicit graph, we could simply mark a state when we visited, and thereby spot repeats (assuming we clear those marks between searches)

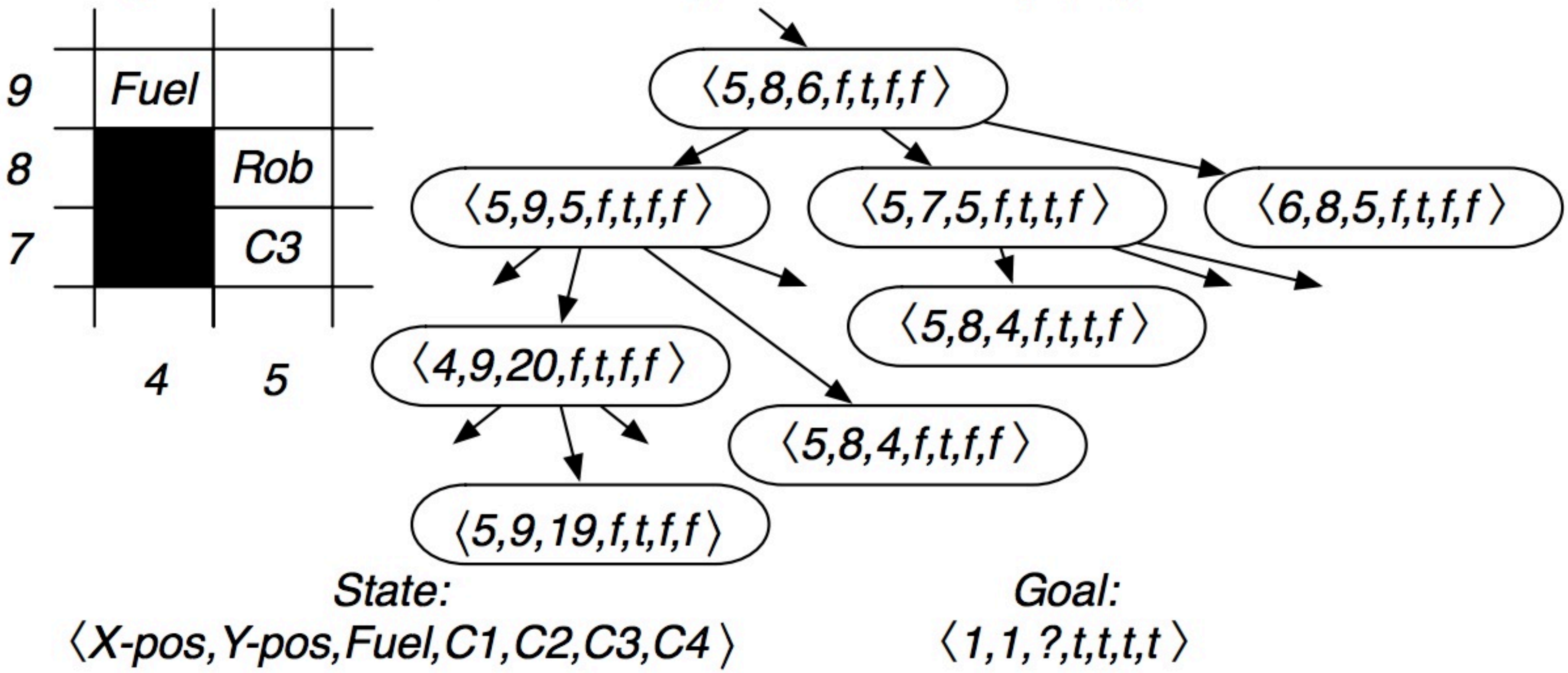
Partial Search Space for a Video Game

Grid game: Rob needs to collect coins C_1, C_2, C_3, C_4, without running out of fuel, and end up at location $(1, 1)$:

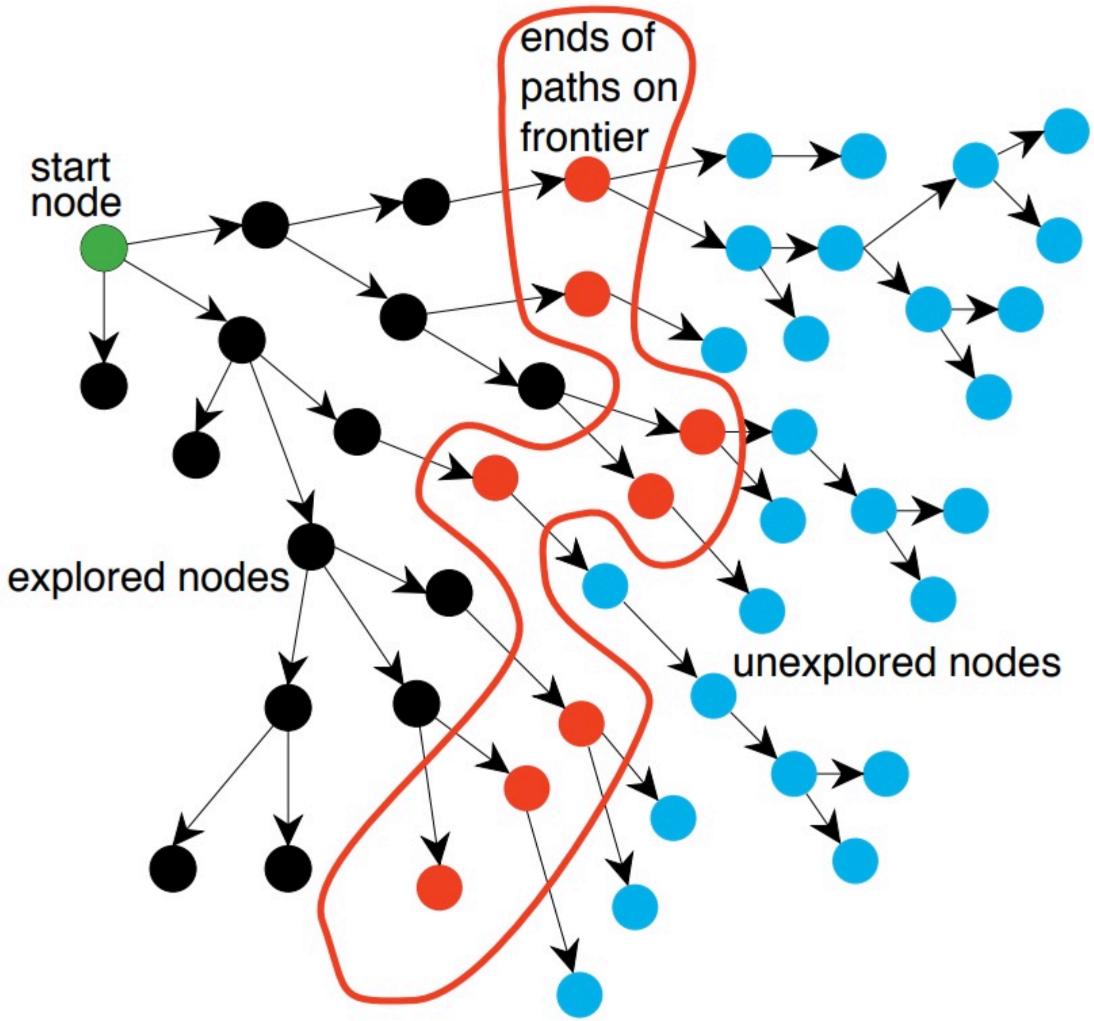


Partial Search Space for a Video Game

Grid game: Rob needs to collect coins C_1, C_2, C_3, C_4, without running out of fuel, and end up at location $(1,1)$:



Problem Solving by Graph Searching

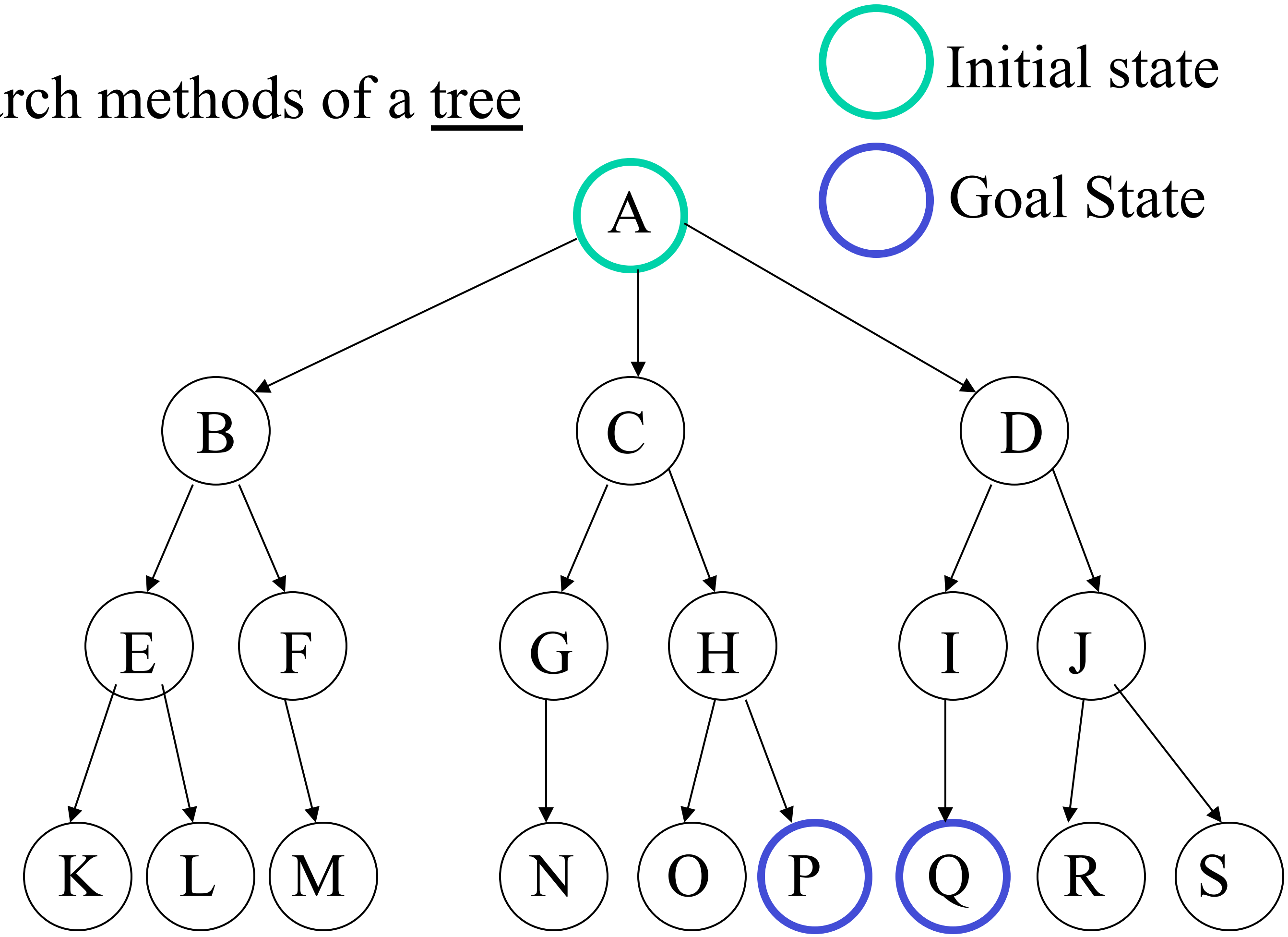


Graph Search Algorithm

Input: a graph,
a set of start nodes,
Boolean procedure $goal(n)$ that tests if n is a goal node.

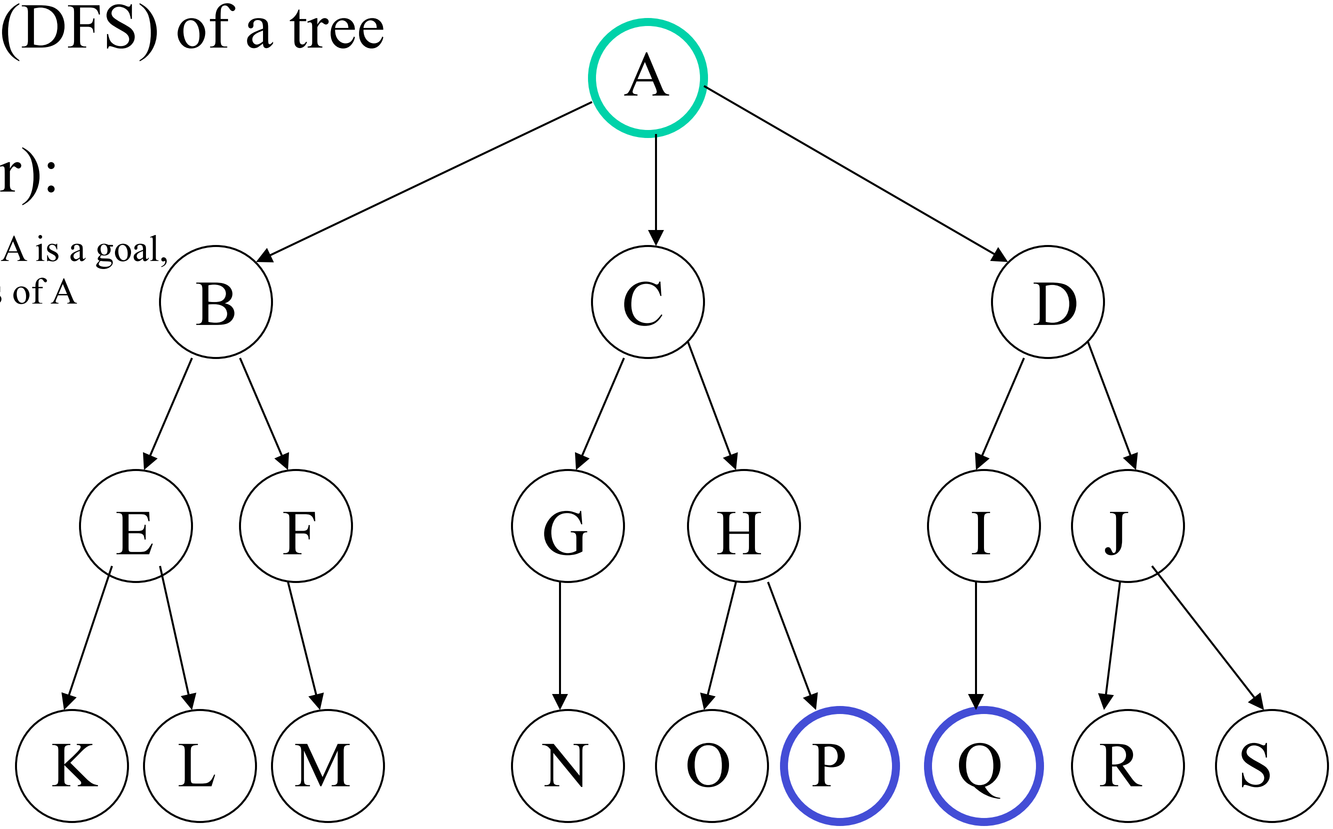
$frontier := \{\langle s \rangle : s \text{ is a start node}\}$
while *frontier* is not empty:
    select and **remove** path $\langle n_0, \ldots, n_k \rangle$ from *frontier*
    if $goal(n_k)$
        return $\langle n_0, \ldots, n_k \rangle$
    for every neighbor n of n_k
        add $\langle n_0, \ldots, n_k, n \rangle$ to *frontier*
end while

Basic Search methods of a tree



Leaves are terminal states with no successors (or neighbors).

Depth-first search (DFS) of a tree



State Stack (Frontier):

{**A**}

Pop A, check if A is a goal,
Push successors of A

{**B**, C, D }

{**E**, F, C, D }

{**K**, L, F, C, D }

{**L**, F, C, D }

{**F**, C, D }

{**M**, C, D }

{**C**, D }

{**G**, H, D}

{**N**, H, D }

{**H**, D }

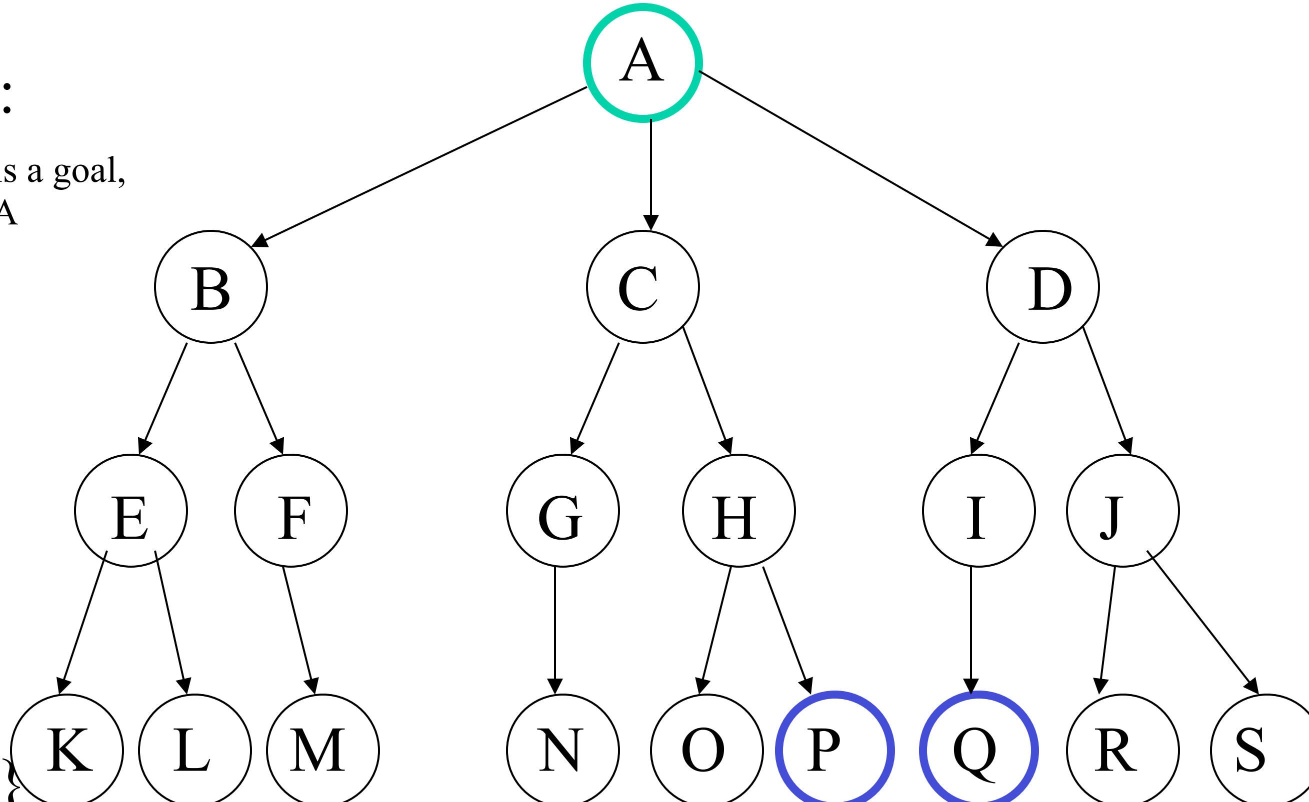
{**O**, P, D}

{**P**, D}

Assume each state in stack also includes the path used to reach it.
So, A is really (A, ( )), B is really (B, (A)), E is really (E, (A B)),
K is really (K, (A, B, E)), L is really (L, (A, B, E)), F is (F, (A, B)), etc.
This convention differs very slightly from textbook's.

Pop P, check if P a goal, stop, return (P, (A, C, H)) or (A→ C → H → P)

Breadth-first search (BFS) of a tree



State Queue (Frontier):

{**A**} Dequeue A, check if A is a goal, Enqueue successors of A

{**B**, C, D }

{**C**, D, E, F }

{**D**, E, F, G, H }

{**E**, F, G, H, I, J}

{**F**, G, H, I, J, K, L}

{**G**, H, I, J, K, L, M}

{**H**, I, J, K, L, M, N}

{**I**, J, K, L, M, N, O, P}

{**J**, K, L, M, N, O, P, Q}

{**K**, L, M, N, O, P, Q, R, S}

{**L**, M, N, O, P, Q, R, S}

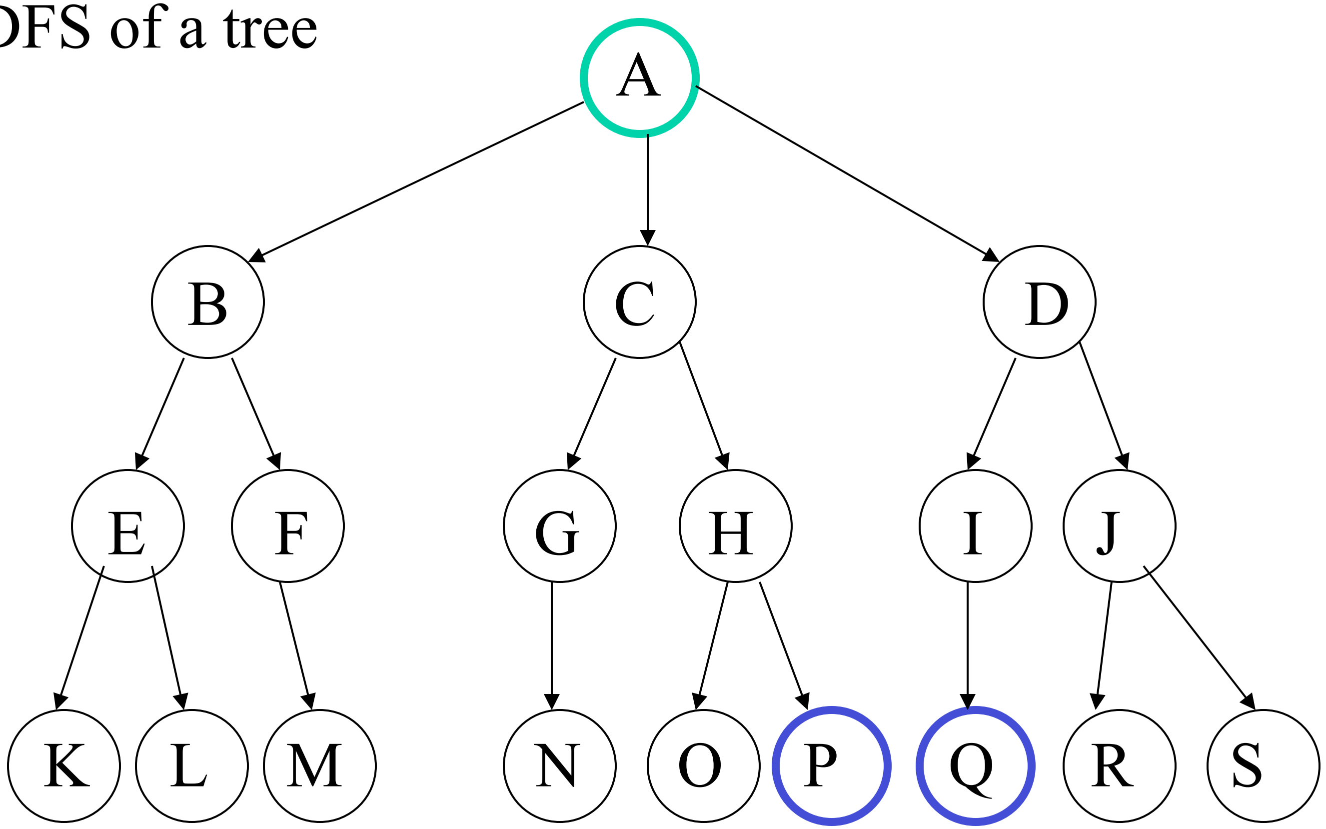
{**M**, N, O, P, Q, R, S}

{**N**, O, P, Q, R, S}

{**O**, P, Q, R, S}

{**P**, Q, R, S} Dequeue P, check if P a goal, stop, return (P, (A, C, H)) or (A → C → H → P)

Iterative deepening DFS of a tree



State Stack (Frontier):

1] {**A**}, {}

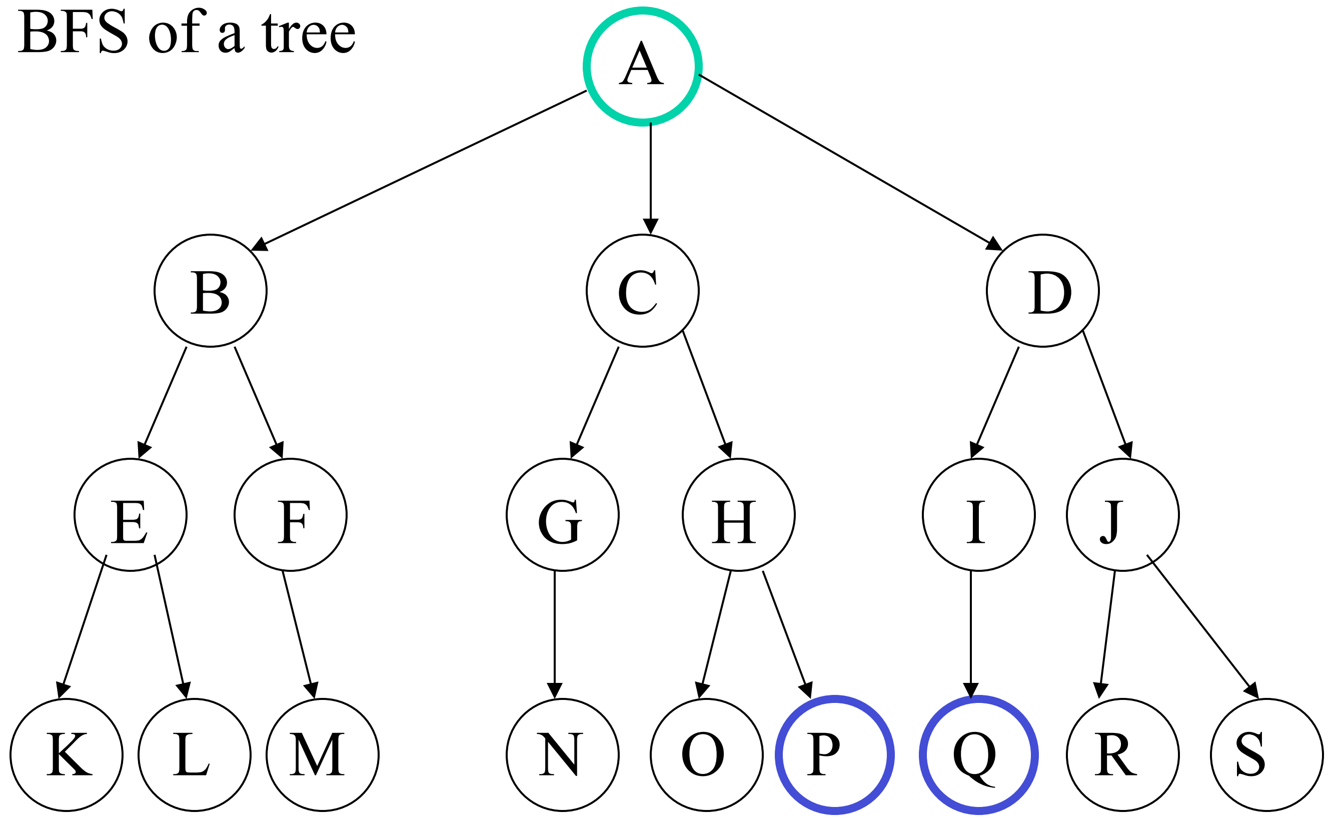
2] {A}, {B, C, D}, {C, D}, {D}, {}

3] {A}, {B, C, D}, {E, F, C, D},{F, C, D}, {C, D}, {G, H, D}, {H, D}, {D}, {I, J}, {J}, {}

4] {A}, {B, C, D}, {E, F, C, D}, {K, L, F, C, D}, {L, F, C, D}, {F, C, D}, {M, C, D}, {C, D}, {G, H, D}, {N, H, D}, {H, D}, {O, P, D}, {P, D}

Pop P, check if P a goal, stop, return (P, (A, C, H)) or (A→ C → H → P)

Iterative widening BFS of a tree



State Queue:

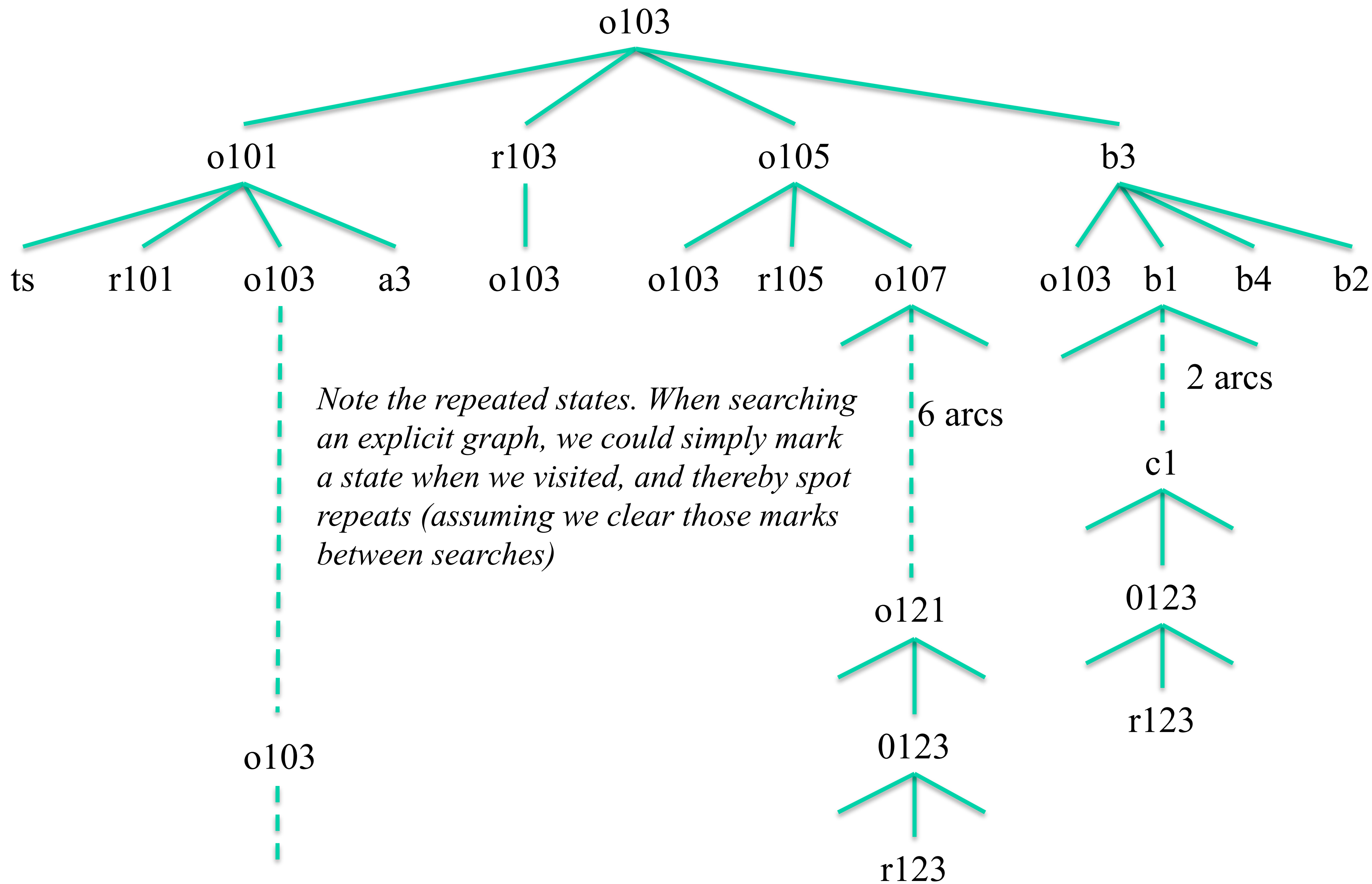
1] {**A**}, {B, ~~C, D~~}, {E}, {K}, {}

2] {**A**}, {B, C}, {C, E}, {E, G}, {G, K}, {K, N}, {N}, {}

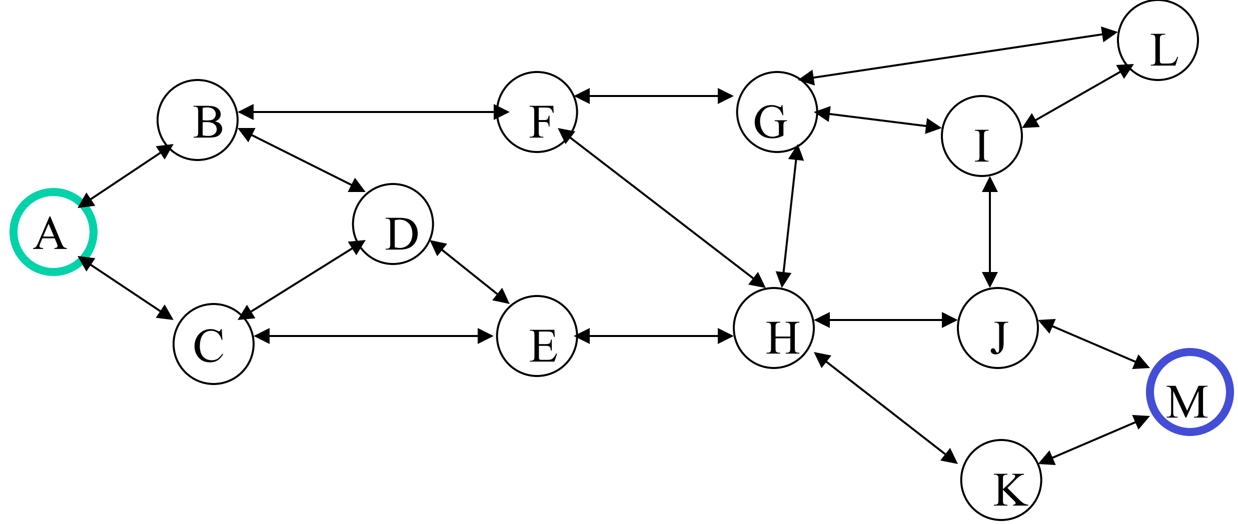
3] {A}, {B, C, D}, {C, D, E}, {D, E, G}, {E, G, I}, {G, I, K}, {I, K, N}, {K, N, Q}, {N, Q}, {Q}

Dequeue Q, check if Q a goal, stop, return (Q, (A, D, I)) or (A → D → I → Q)

Search tree for Figure 3.1 of Poole and Mackworth, assuming robot starts at 0103, with a goal of being in r123, with all arcs bidirectional (a different assumption than text's)



Searching a graph with DFS

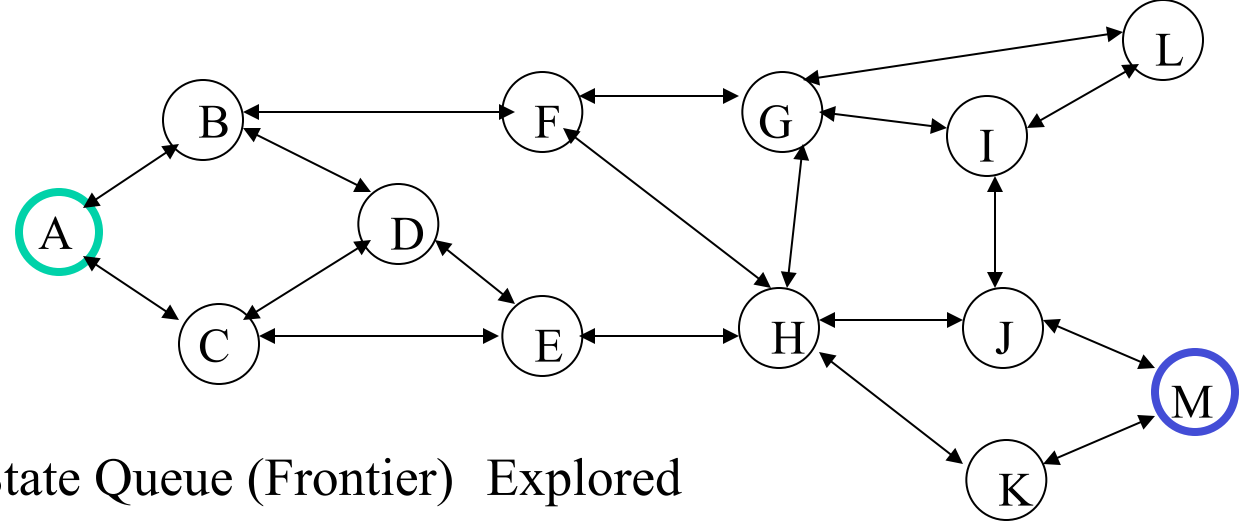


Successors not replaced on Open (already on Open or Explored)	Path to TOS State	State Stack (Frontier)	Explored States
		{A}	{}
	(A)	{B, C}	{A}
A	(A,B)	{F, D, C}	{A, B}
B	(A,B,F)	{G, H, D, C}	{A, B, F}
F H	(A,B,F,G)	{L, I, H, D, C}	{A, B, F, G}
G I	(A,B,F,G)	{I, H, D, C}	{A, B, F, G, L}
G L	(A,B,F,G,I)	{J, H, D, C}	{A, B, F, G, L, I}
H I	(A,B,F,G,I,J)	{M, H, D, C}	{A, B, F, G, L, I, J}

M is goal, return (M, (A,B,F,G,I,J))

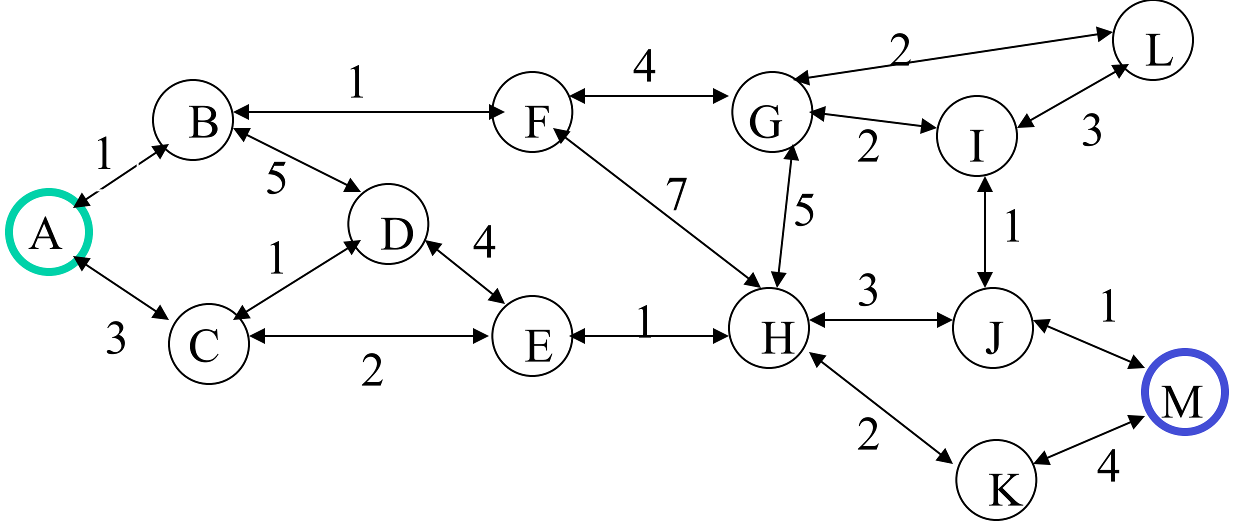

Tennessee Supreme Court
h (Westend via Kessington)
h (25th Ave Garage by Jess Neely)
Westend via Kessington
Memorial Gym
Jess Neely and 25th Ave Garage

Searching a graph with BFS



Repeated Successors	Path	State Queue (Frontier)	Explored
	( )	{A}	
	(A)	{B,C}	{A}
A	(A)	{C,D,F}	{A,B}
A,D	(A,B)	{D,F,E}	{A,B,C}
B,C,E	(A,B)	{F,E}	{A,B,C,D}
B	(A,C)	{E,G,H}	{A,B,C,D,F}
C,D,H	(A,B,F)	{G,H}	{A,B,C,D,F,E}
F,H	(A,B,F)	{H,L,I}	{A,B,C,D,F,E,G}
E,F,G	(A,B,F,G)	{L,I,J,K}	{A,B,C,D,F,E,G,H}
G,I	(A,B,F,G)	{I,J,K}	{A,B,C,D,F,E,G,H,L}
G,L,J	(A,B,F,H)	{J,K}	{A,B,C,D,F,E,G,H,L,I}
H,I	(A,B,F,H)	{K,M}	{A,B,C,D,F,E,G,H,L,I,J}
H,M	(A,B,F,H,J)	{M}	M is goal, return (M, (A,B,F,H,J))

Searching a graph with costs (Lowest Cost First Search)



Repeated Successors	Path	State Priority Queue (Frontier)	Closed
	( )	{A(0)}	
	(A)	{B(1), C(3)}	A(0)
A(1)	(A,B)	{F(2), C(3), D(6)}	A(0),B(1)

continue the example