ECE 457A ADAPTIVE COOPERATIVE ALGORITHMS

LOCAL SEARCH Definitions and Strategies

Recall

- Which of the following are examples of ill-structured problems?
  - ❑Converting temperature from Celsius to Fahrenheit
  - ❑Increasing water supply for a growing community
  - ❑Calculating speed given distance and time
  - ❑Designing a new underground subway system that spans multiple countries
  - ❑Maximizing the efficiency of a manufacturing process
  - ❑Scheduling a NHL season to maximize viewing
- Which of the following aspects are **not** well defined in an ill-structured problem?
  - ❑Goals
  - ❑Beginning state
  - ❑Actions
  - ❑End state
  - ❑Constraints

SEARCH TREE - TERMINOLOGY

We search for the best state possible:

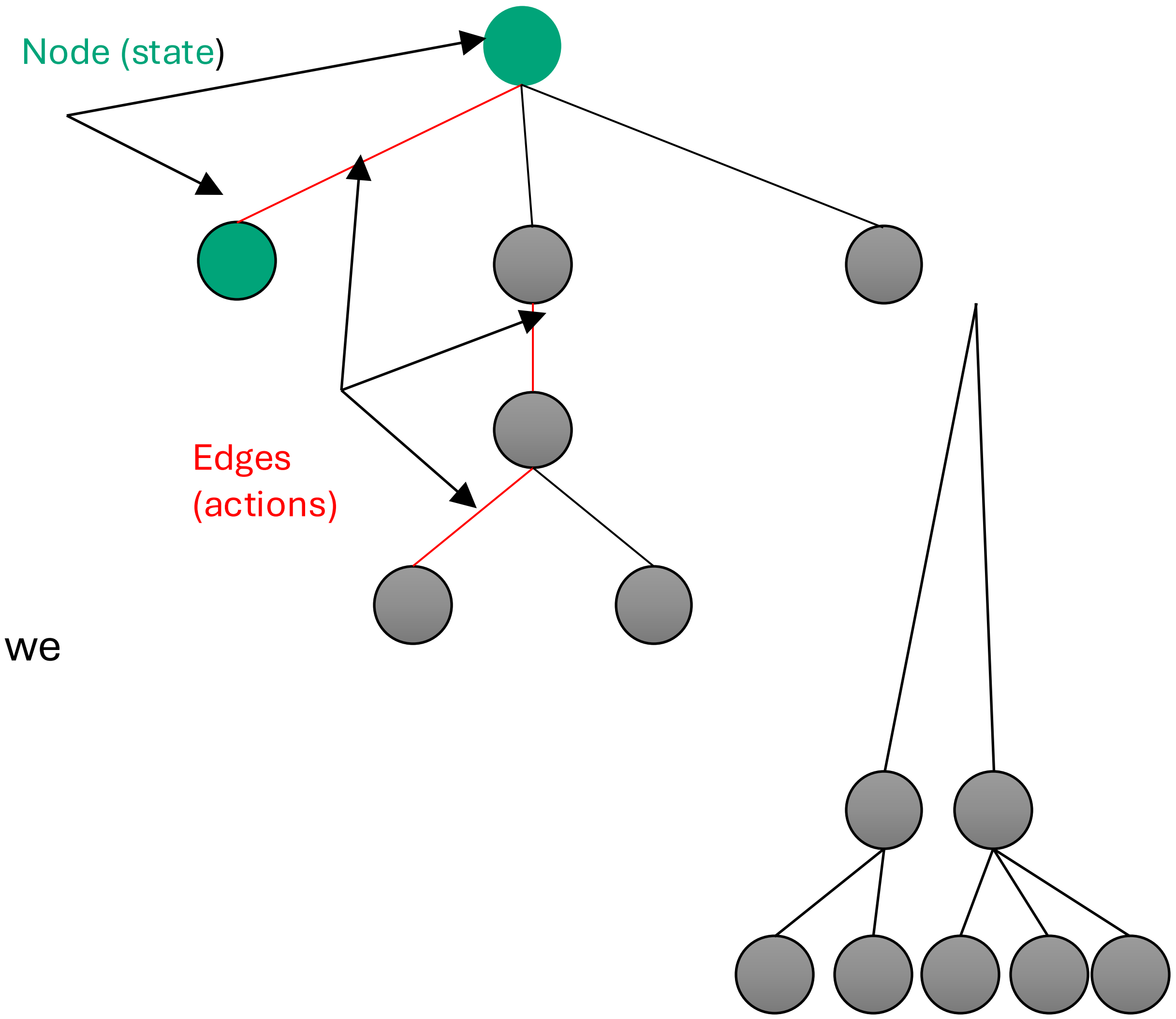
Problem graph representation

Node=State

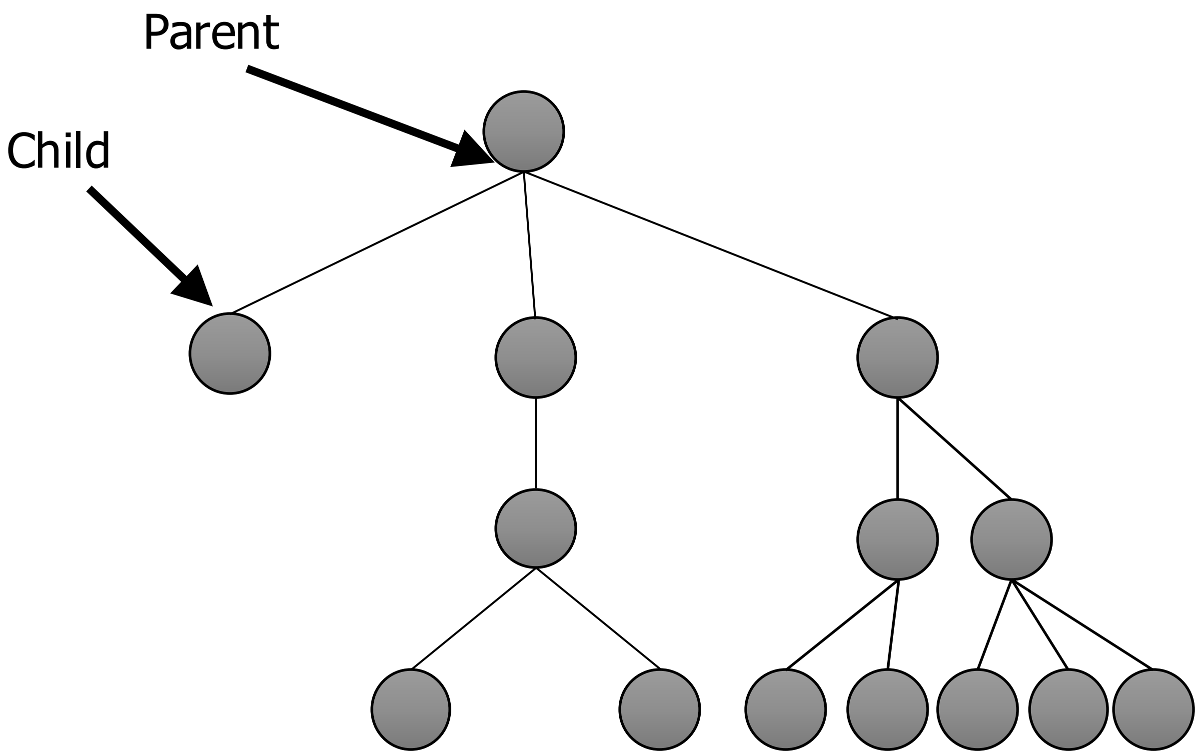
- Optimal state: nothing better.
- Suboptimal state: Good enough
- When at a state, we evaluate its merit wrt to the goal we are searching for: as a goal state itself, or as a step forward towards our goal.

Edge=Action

- An Edge tells us two things:
    1. The states that we can reach/materialize in one move from the state we are at- by taking the action associated with edge.
    2. A labeled edge signifies a cost/gain associated with the action. No label implies unit cost/gain.

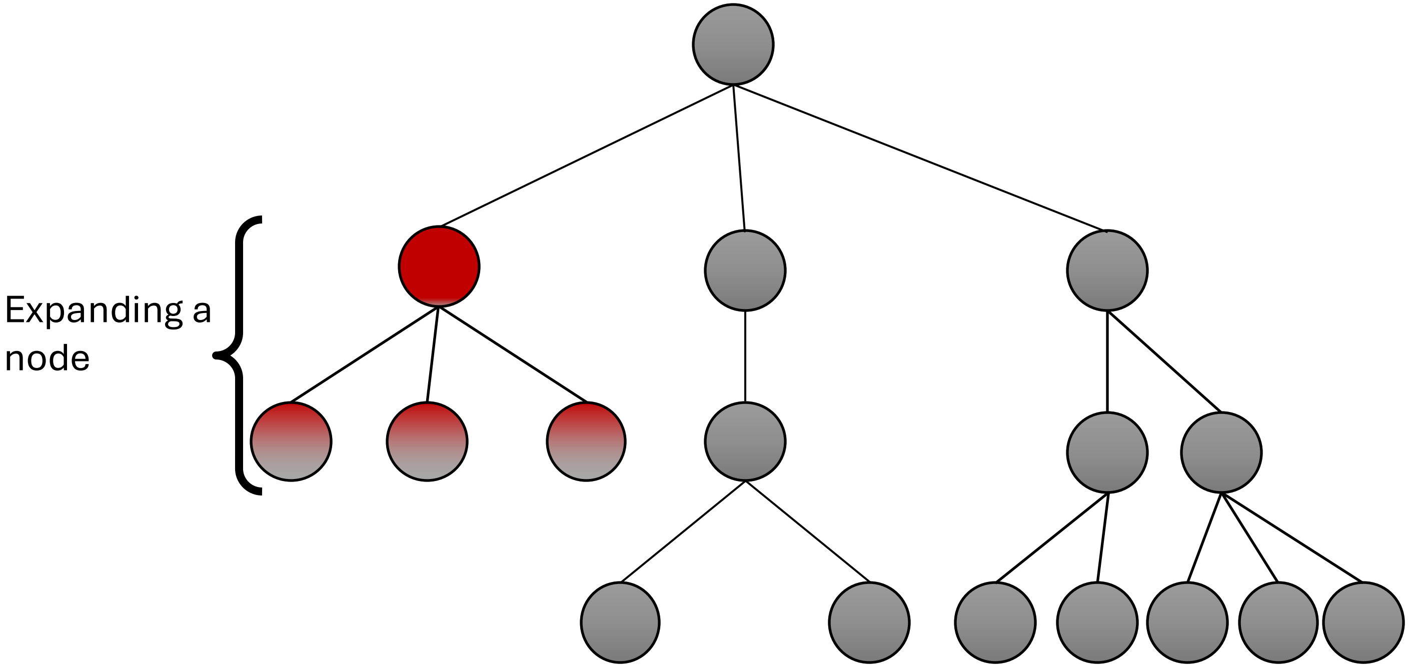


SEARCH TREE - TERMINOLOGY

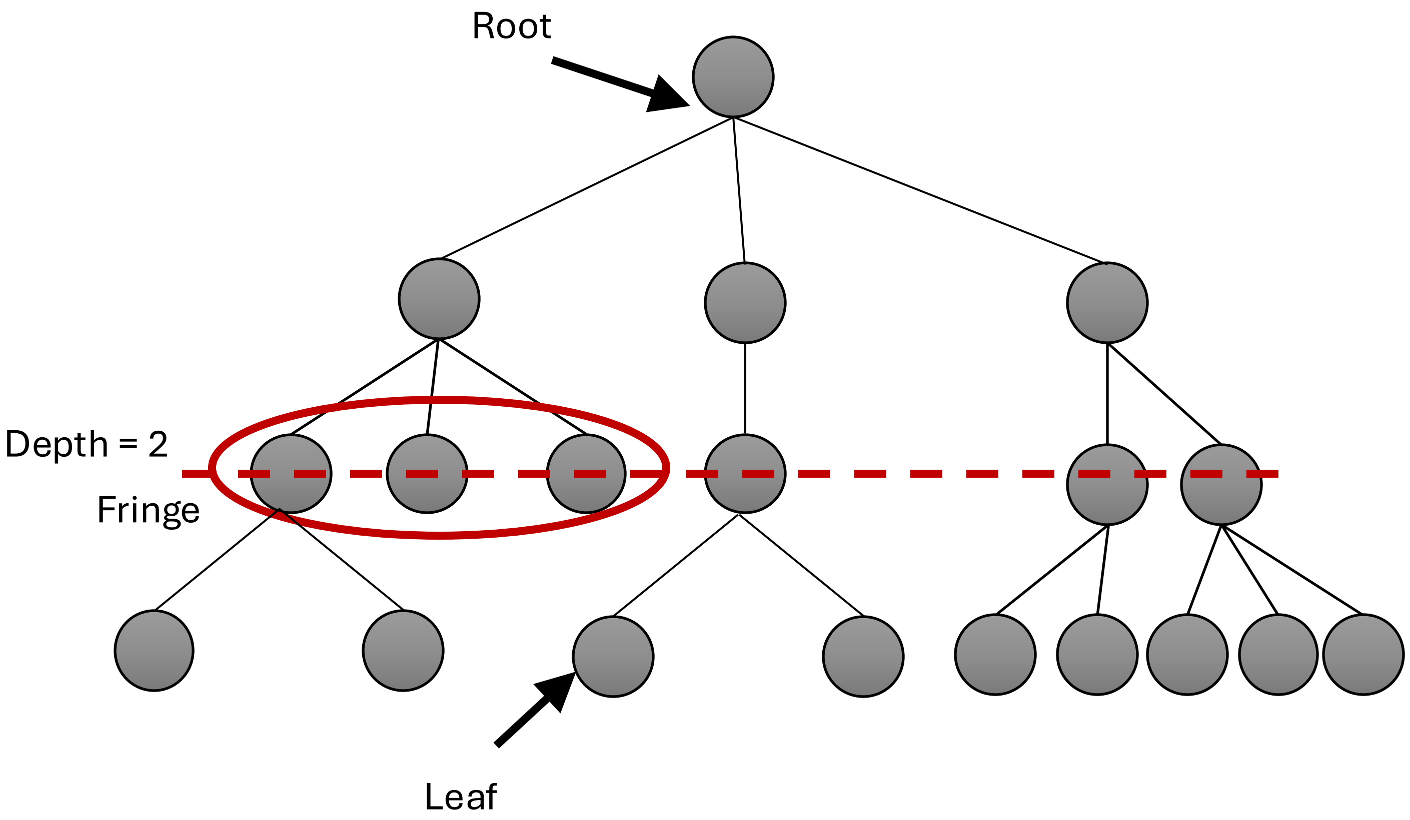


Search Tree - Terminology

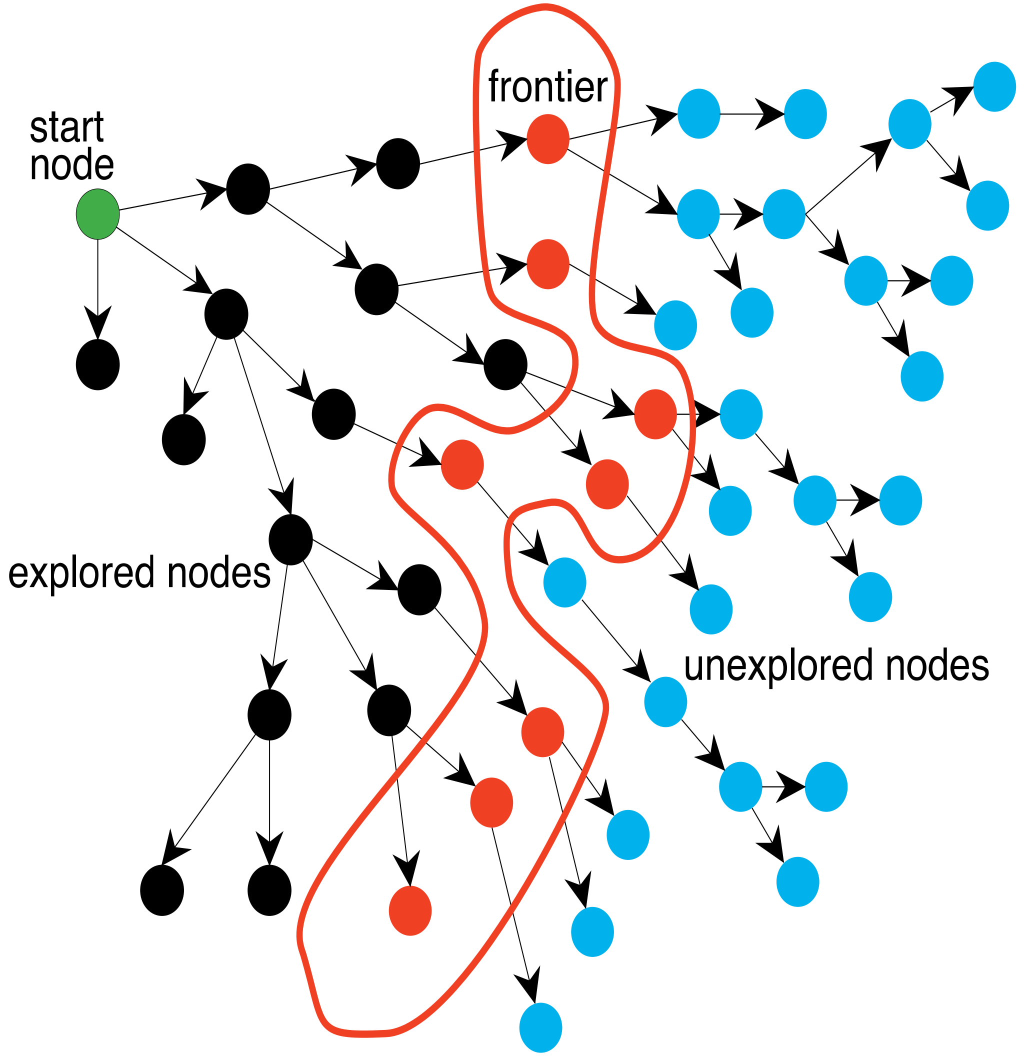
We perform our search as a series of node (state) expansions



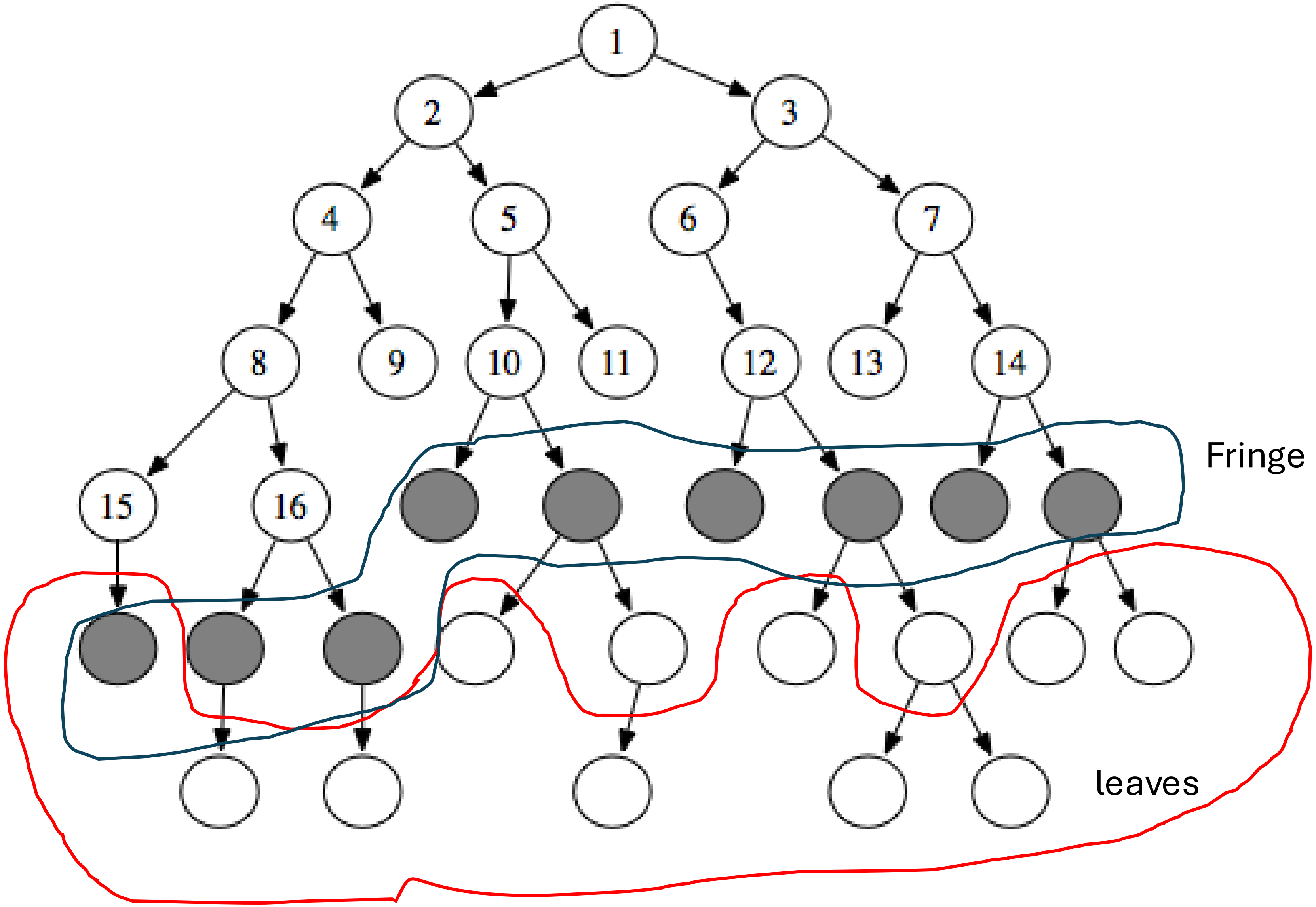
SEARCH TREE - TERMINOLOGY



States: ARE EITHER EXPLORED, AT THE FRONTIER, OR UNEXPLORED

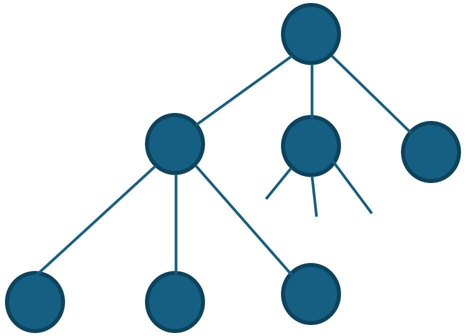


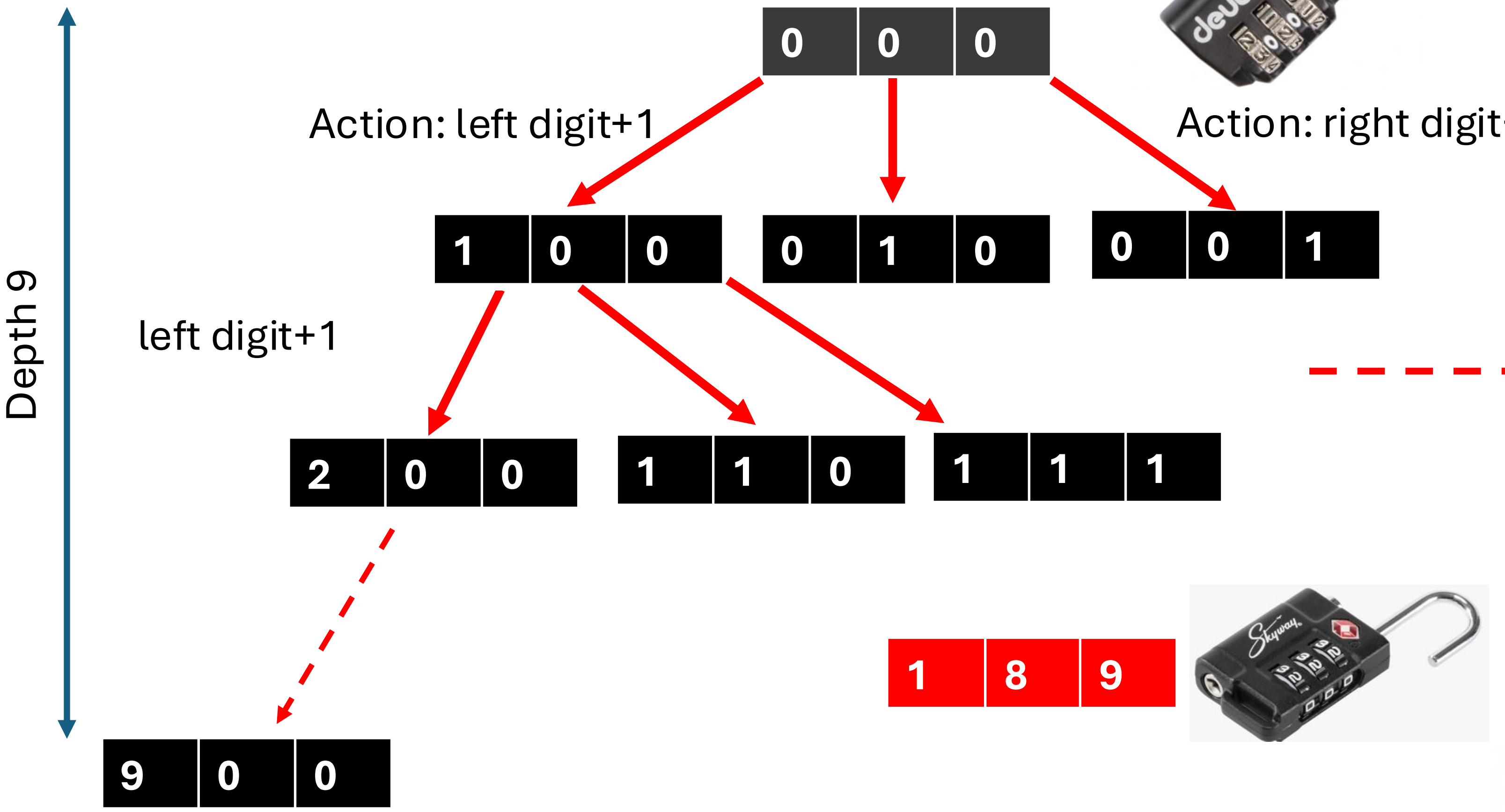
LEAVES VS FRINGE



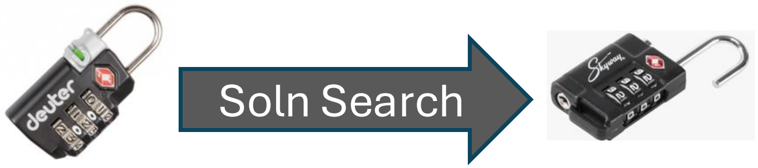
SIMPLE EXAMPLE

Branching factor:3
Depth: 9
Search space:





0 0 9



The Locker unlocking problem.

Search strategy performance attributes

Complete search strategy

Definition (complete)

If a solution exists, a complete algorithm is guaranteed to find a solution within a finite amount of time.

Optimal search strategy

Definition (optimal)

If a solution exists and an algorithm finds a solution, then the first solution found by an optimal algorithm is the solution with the lowest cost.

SEARCH STRATEGY PERFORMANCE ATTRIBUTES

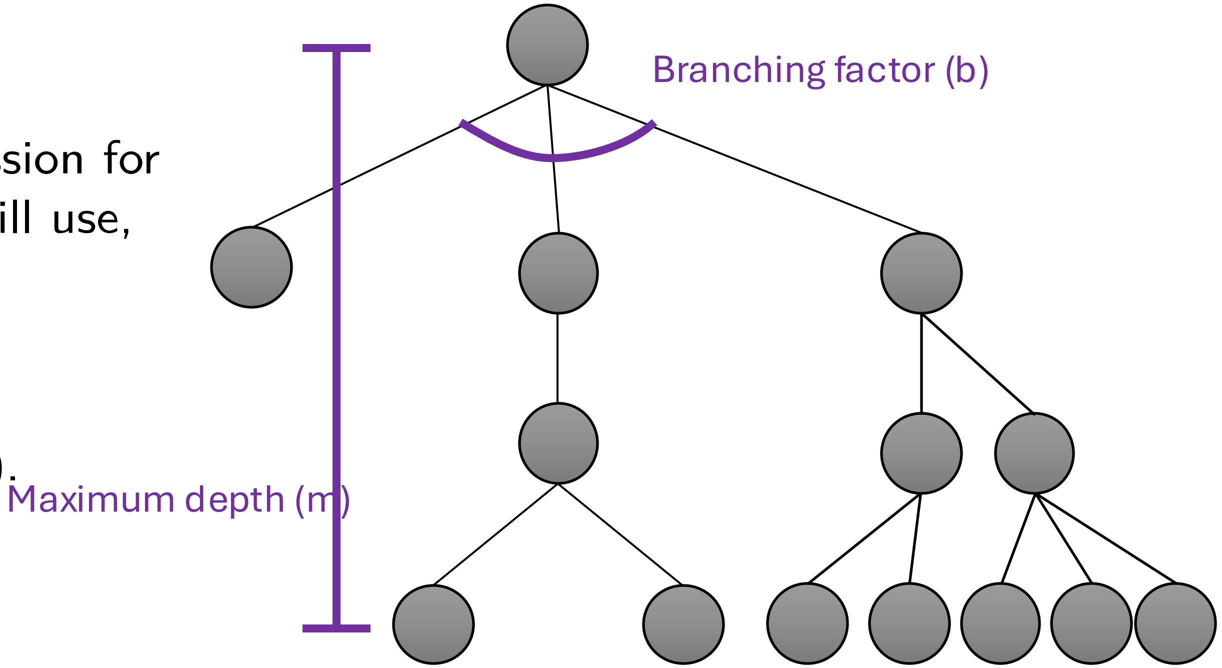
Memory requirement growth

Definition (space complexity)

The space complexity of a search algorithm is an expression for the worst-case amount of memory that the algorithm will use, expressed in terms of *b*, *d*, and *m*.

Useful definitions:

- *b*: the maximum branching factor (may be infinite).
- *d*: the depth of the shallowest goal node (finite).
- *m*: the maximum path length (may be infinite).



Time requirement growth

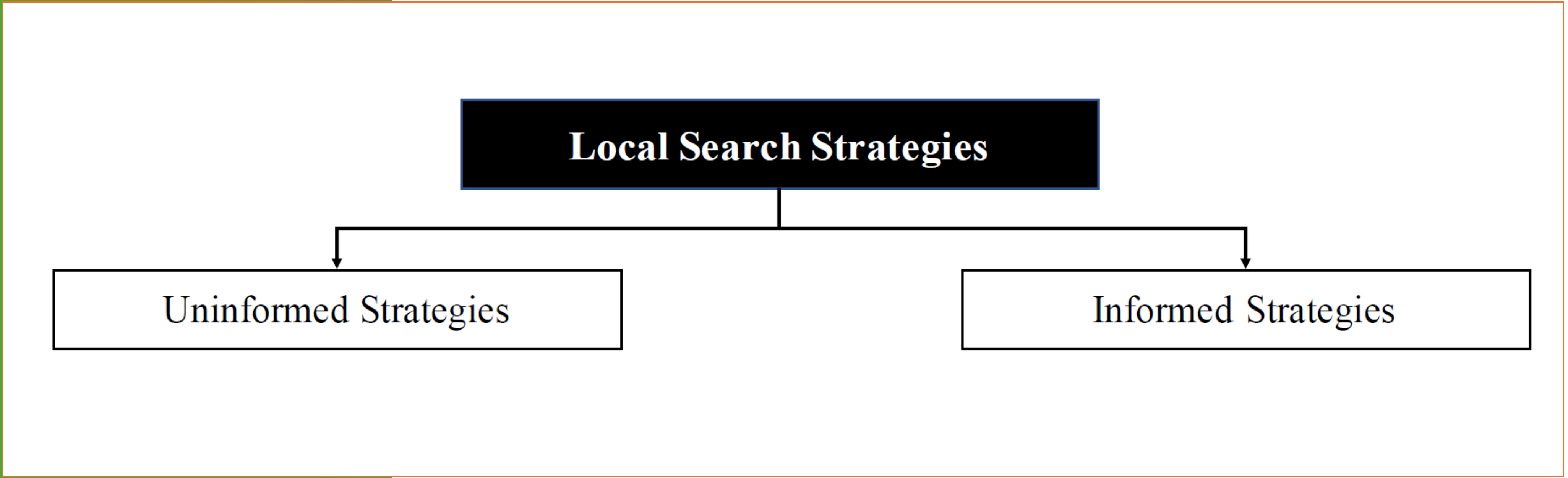
Definition (time complexity)

The time complexity of a search algorithm is an expression for the worst-case amount of time it will take to run, expressed in terms of *b*, *d*, and *m*.

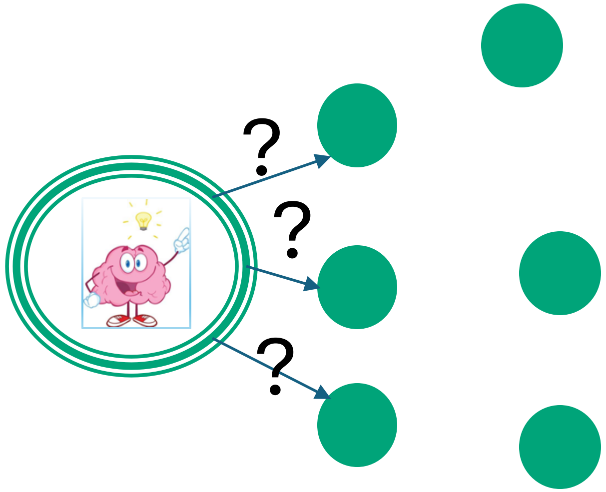
Local Search strategies

How they operate and cost implications.

Types of Local Search strategies



Template of Generic Search



Given a graph representation of the problem, we construct a search tree,

A generic search on the solution tree is a repetition of **choose a state, test the state, and expand the state.**

A particular search strategy influences how we choose the **next node (state) to consider** in the search! (this is where search strategies differ).

Generally, a **queue** structure is used to store nodes on the **fringe** to be expanded.

Different search strategies use different data structures.

Uniformed/blind Search

Uninformed/blind Search Strategies

- Also known as "***blind search***," uninformed search strategies use **no information** about the likely "direction" of the goal node(s)
- Only has the information provided by the problem formulation (initial state, set of actions, goal test, cost)

- Uninformed search methods:
  - breadth-first,
  - depth-first,
  - depth-limited,
  - uniform-cost,
  - depth-first iterative deepening,
  - bidirectional

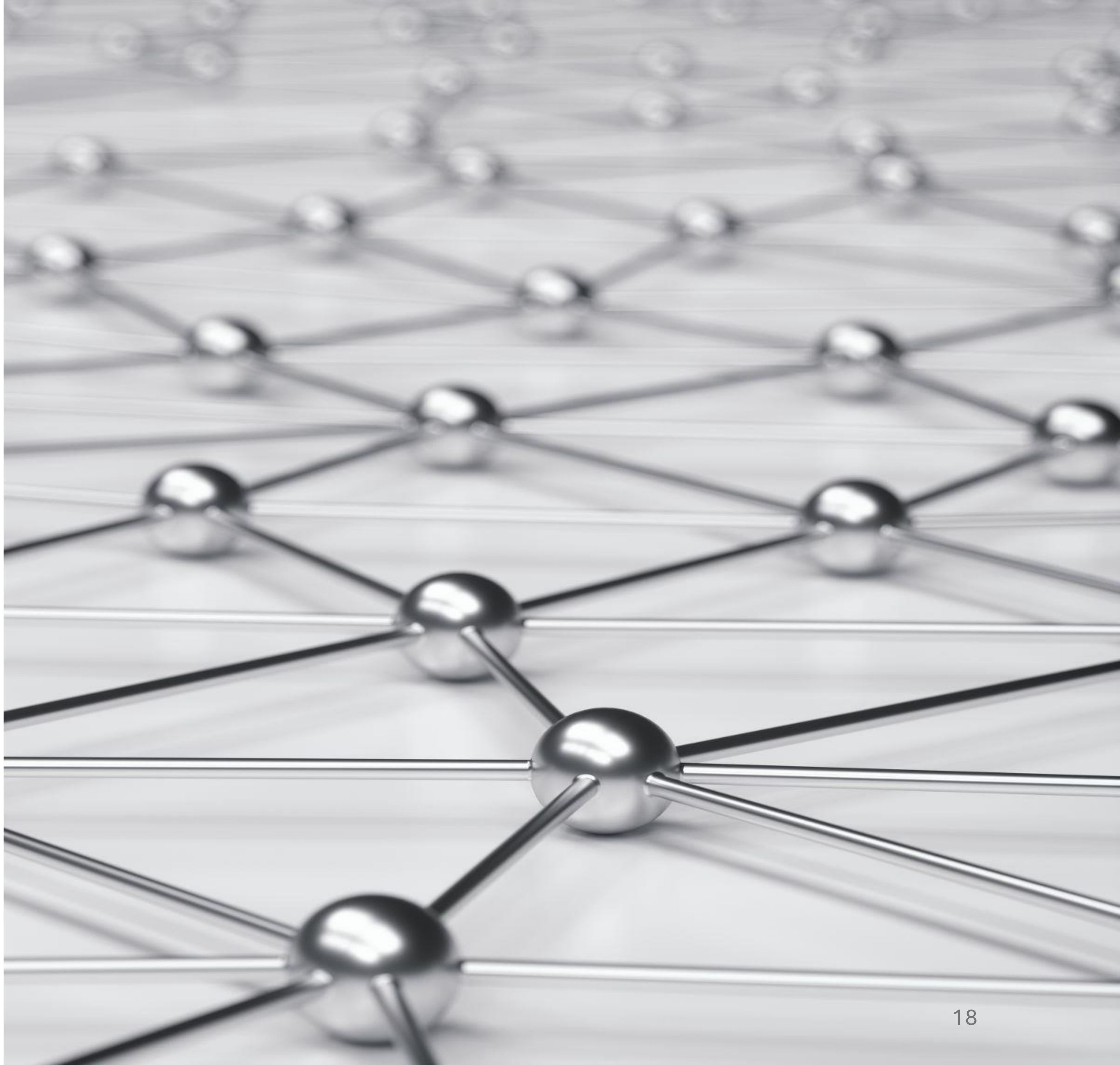
RECAL: Types of Search

• Informed Search

- Also known as “heuristic search,” informed search strategies use information about the domain to head in the general direction of the goal node(s)
- Has additional information that allows it to judge the promise of an action, i.e. the estimated cost from a state to a goal
- Informed search methods:
  - Hill climbing,
  - best-first,
  - greedy search,
  - beam search, A, A*

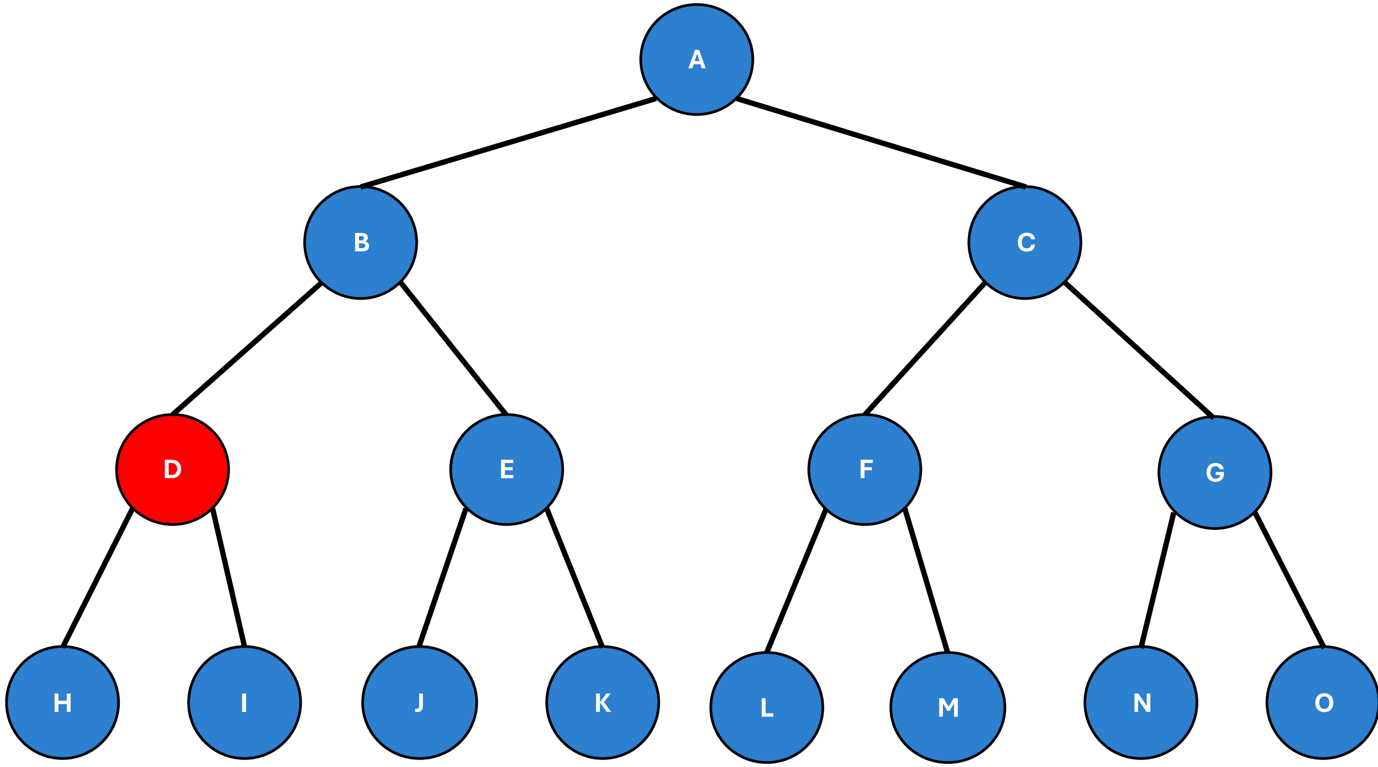
Breadth-First Search

Uninformed search



Breadth-First Search

Problem: Search for state D



Search Strategy: Breadth-first search

Search through a tree one level at a time.

- We traverse through one entire level of children nodes first, before moving on to traverse through the grandchildren nodes.
- And we traverse through an entire level of grandchildren nodes before going on to traverse through great-grandchildren nodes.

Breadth-First Search

- Mechanics:
- Expand the ***shallowest unexpanded node first***
- Fringe: nodes waiting in the queue to be expanded

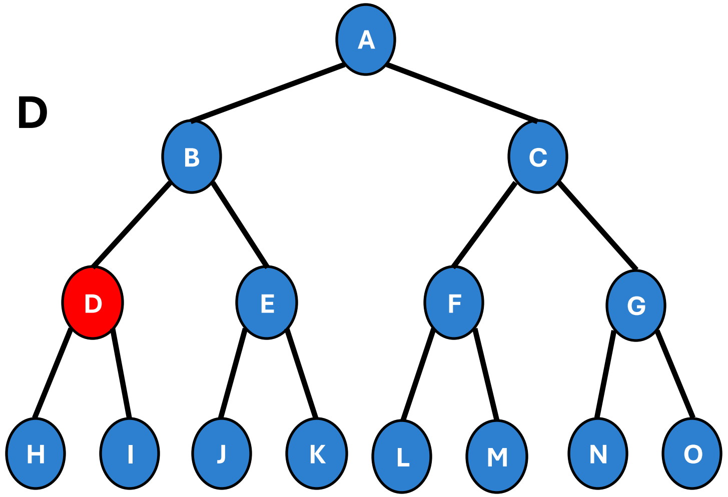
  - Fringe is a ***FIFO*** queue (first-in, first-out)
  - New successors go in the end of the queue

Initial State

A

<< Is this the goal state?

Problem: Search for state D



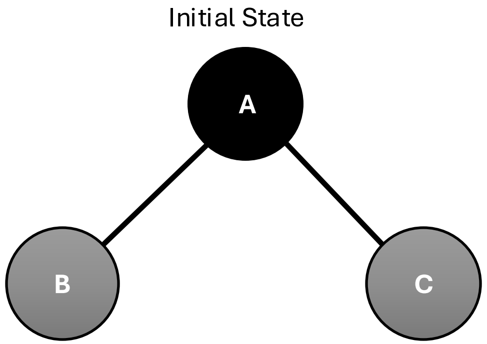
A

Fringe queue

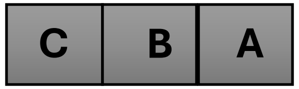
Breadth-First Search

Breadth First strategy:

- Expand the shallowest unexpanded node first
- New successors go in the end of the queue



Is this the goal state? >>

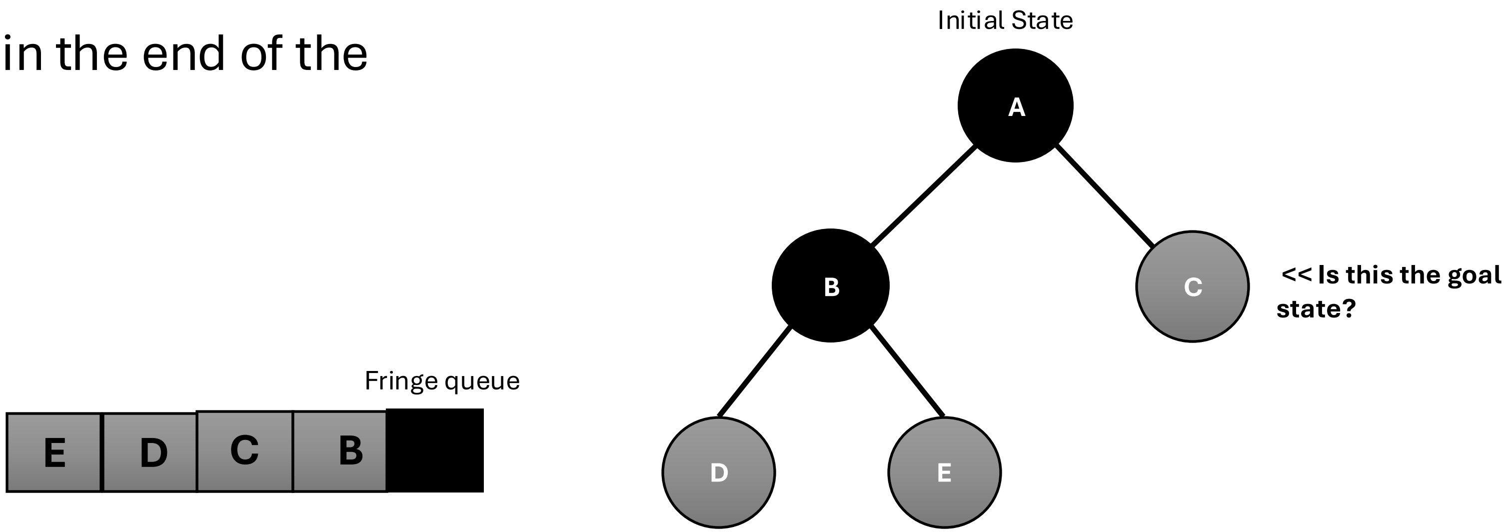


Fringe queue

Breadth-First Search

Breadth First strategy:

- Expand the shallowest unexpanded node first
- New successors go in the end of the queue

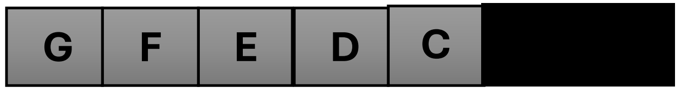


Breadth-First Search

Breadth First strategy:

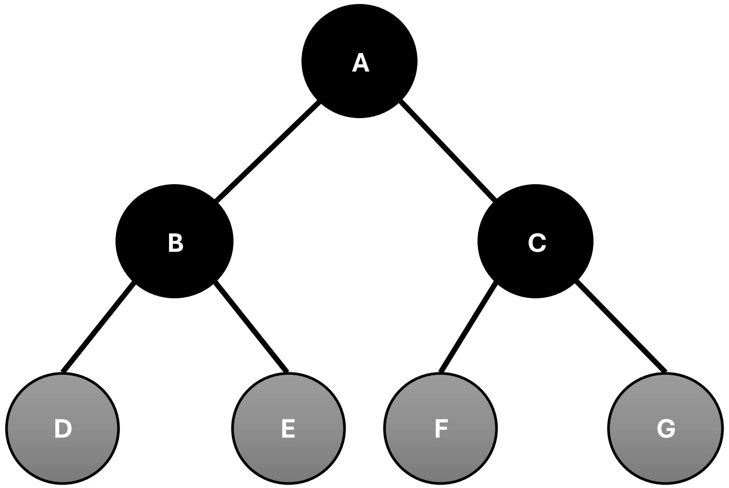
- Expand the shallowest unexpanded node first
- New successors go in the end of the queue

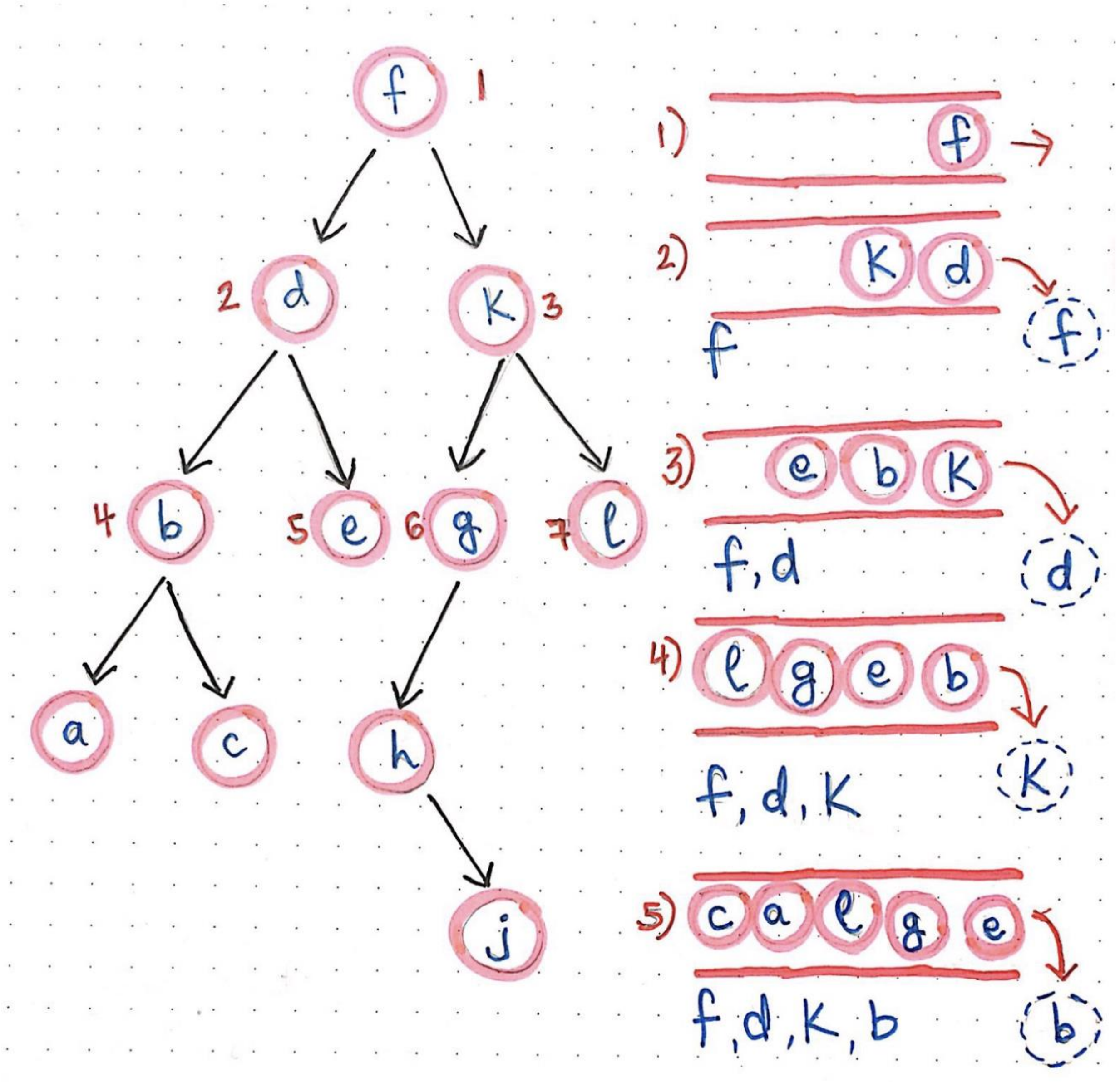
Fringe queue



Is this the goal state? >>

Initial State





f 1
2 d
K 3
4 b
5 e
6 g
7 l
a
c
h
j
1) f
2) K d f
f
3) e b K d
f, d
4) l g e b K
f, d, K
5) c a l g e b
f, d, K, b

BFS Algorithm

BREADTH-FIRST-SEARCH

Algorithm 2 Breadth-First Search

```
put the start state in the frontier
while frontier is not empty do
    remove the oldest node added to the frontier
    if the node contains a goal state then
        return the solution
    end if
    generate all the successors of the node
    add every successor of the node to the frontier
end while
return failure
```

Breadth-First Search

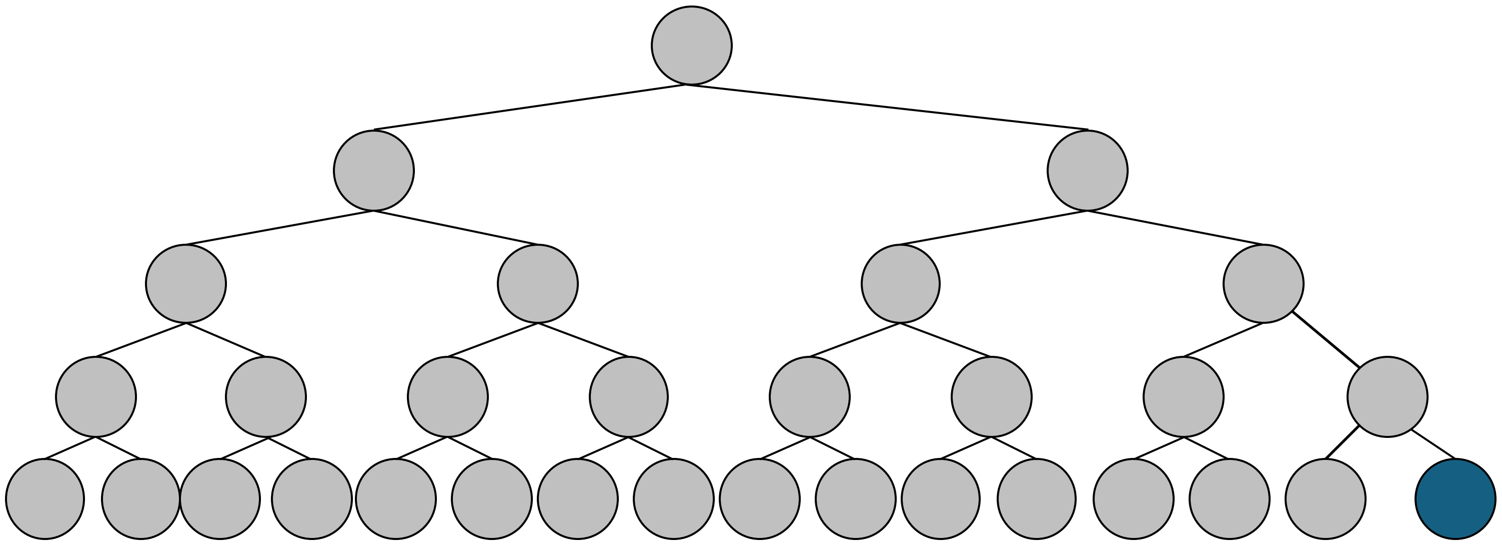
- Upper-bound case: goal is last node of depth d
  - Number of generated nodes:
  - Space & time complexity: all generated nodes

<u>We generate</u>

at d=0) b^0 =1+
at d=1) b =2+
at d=2) b^2 =4+
at d=3) b^3 =8+
at d=4) b^4 =16
at d+1) b^5 =32=(16+8+4+2+1)+1

Total of states generated

4



No of states = $1+b+b^2+b^3+\ldots+b^d = b^{d+1} - 1$
$= O(b^{d+1})$

Breadth-First Search

Another formula:

We generate

at d=0) 1+

at d=1) b =2+

at d=2) b^2 =4+

at d=3) b^3 =8+

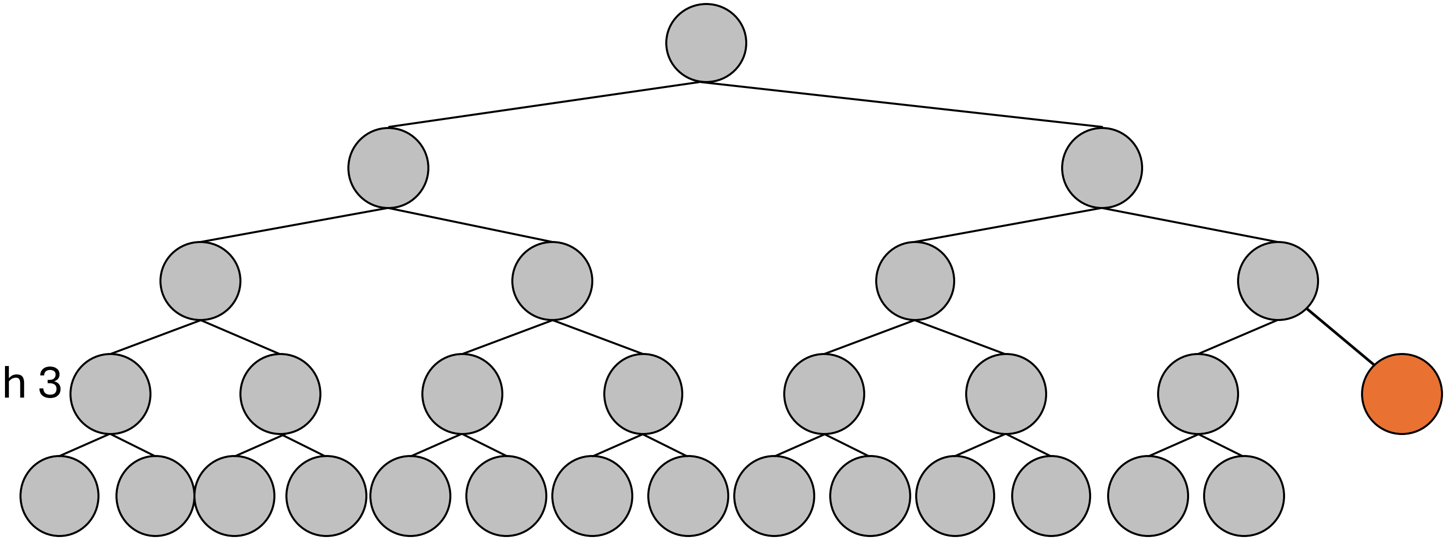
plus we would have generated states at d=4) b^4 =16-b

at d+1) b^5 =32=(16+8+4+2+1)+1

Total No of states=$1+b+b^2+b^3+\ldots+b^d + b^{d+1} - b$

$=O(b^{d+1})$

Goal at depth 3



Breadth-First Search

- **Complete**, if b is finite
- **Optimal**, if path ***cost is equal to depth*** (if all operators have the same cost)
  - Guaranteed to return the shallowest goal (depth d)
- **Exponential time and space complexity**
  - Time complexity = $O(b^{d+1})$
  - Space complexity = $O(b^{d+1})$

  ➢ where d is the depth of the solution and b is the branching factor (i.e., number of children) at each node

COMPLEXITY
TIME AND MEMORY

The costs, however, are very high. Let b be the maximal branching factor and d the depth of a solution path. Then the maximal number of nodes expanded is

$$b + b^2 + b^3 + \ldots + b^d + (b^{d+1} - b) \in O(b^{d+1})$$

Example: $b = 10$, 10,000 nodes/second, 1,000 bytes/node:

Depth	Nodes	Time	Memory
2	1,100	.11 seconds	1 megabyte
4	111,100	11 seconds	106 megabytes
6	10^7	19 minutes	10 gigabytes
8	10^9	31 hours	1 terabyte
10	10^{11}	129 days	101 terabytes
12	10^{13}	35 years	10 petabytes
14	10^{15}	3,523 years	1 exabyte