CSC 384
Introduction to Artificial Intelligence

Uninformed Search

Alice Gao and Randy Hickey

Winter 2023

Learning Goals

By the end of this lecture, you should be able to

- Trace the execution of and implement Breadth-first search, Depth-first search, Iterative-deepening search, and Uniform-cost search.
- Explain the properties of Breadth-first search, Depth-first search, Iterative-deepening search, and Uniform-cost search.
- Describe strategies for pruning the search space.

Outline

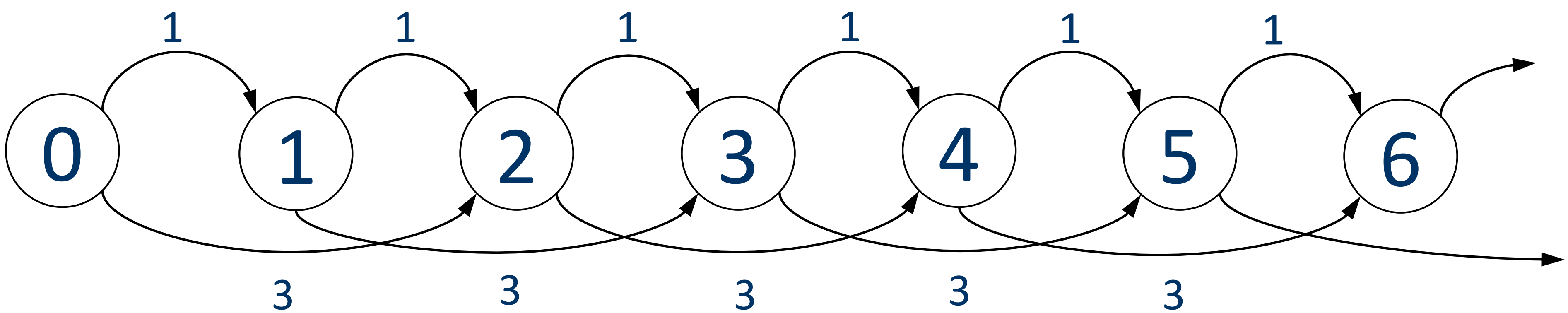
- Generic Search Algorithm
- Depth-First Search
- Breadth-First Search
- Iterative-Deepening Search
- Uniform-Cost Search
- Multiple-Path Pruning

GENERIC SEARCH ALGORITHM

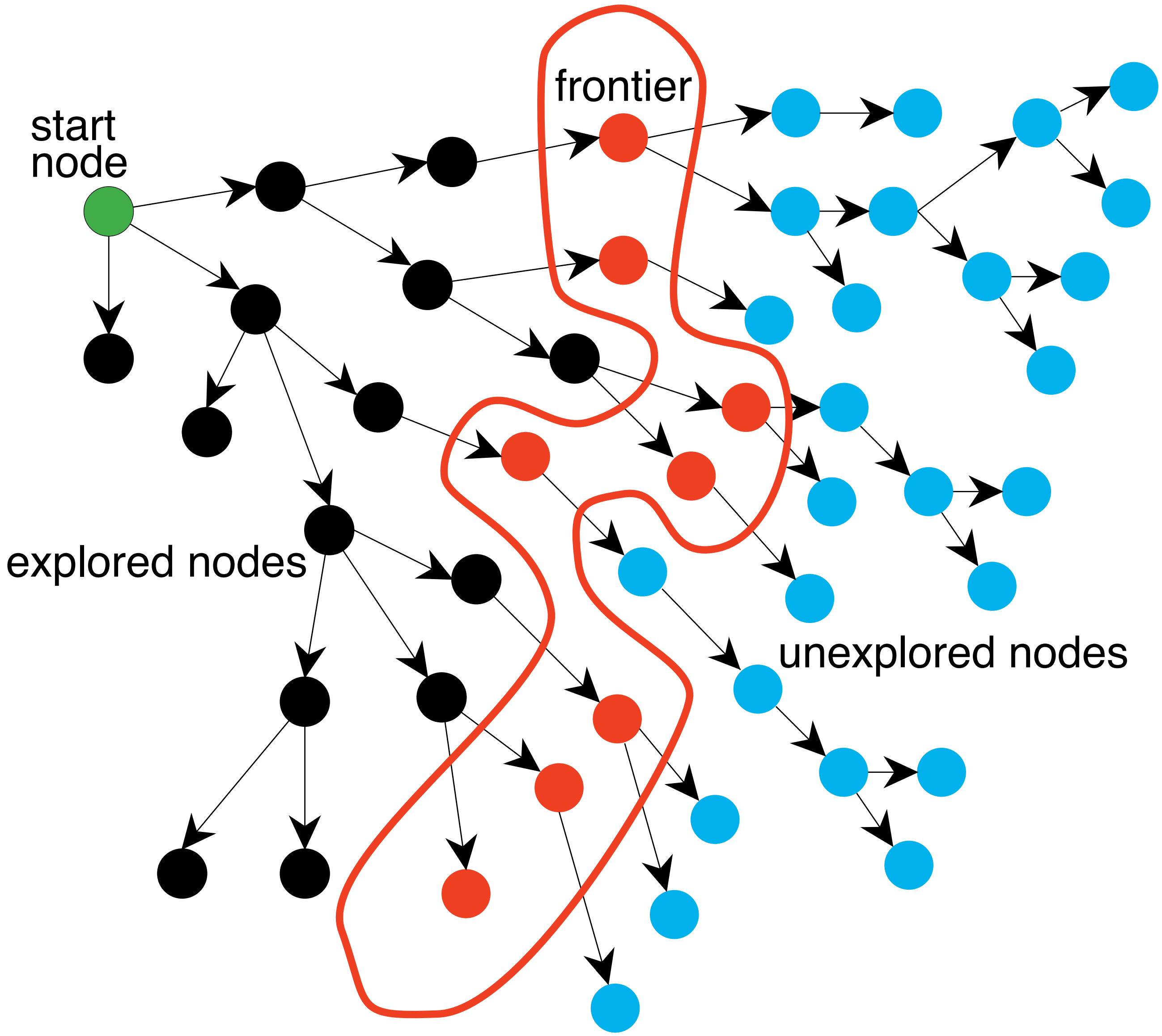
Integer Example

- States: the non-negative integers $\{0, 1, 2, \ldots\}$.
- Initial state: 0.
- Goal state: 5.
- Successor function: $S(n) = \{n + 1, n + 2\}$.
- Cost function: $C(n, n + 1) = 1, C(n, n + 2) = 3$.

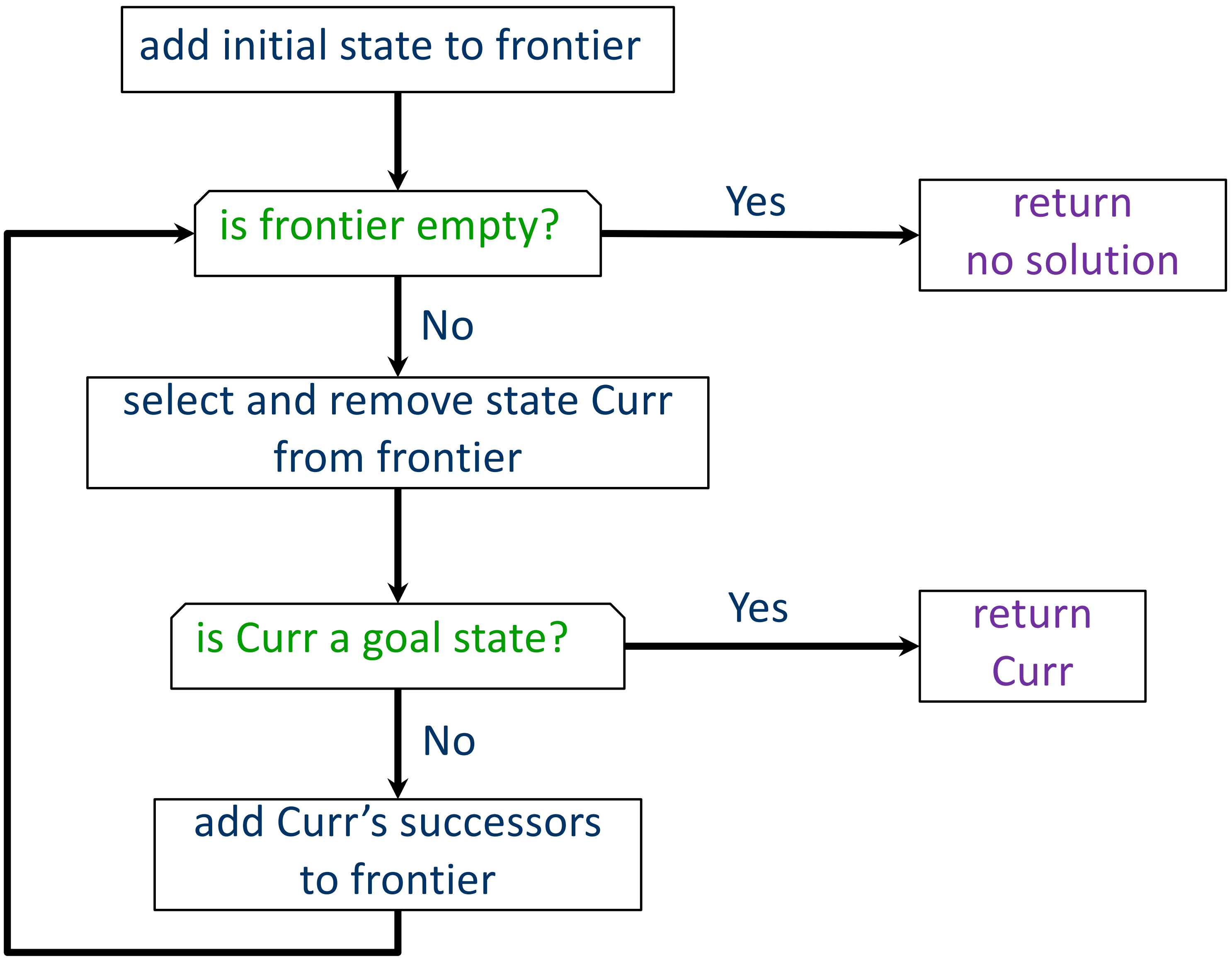
Search Graph for Integer Example



Generic Search Tree



Generic Search Algorithm as a Flowchart



Generic Search Algorithm Pseudocode

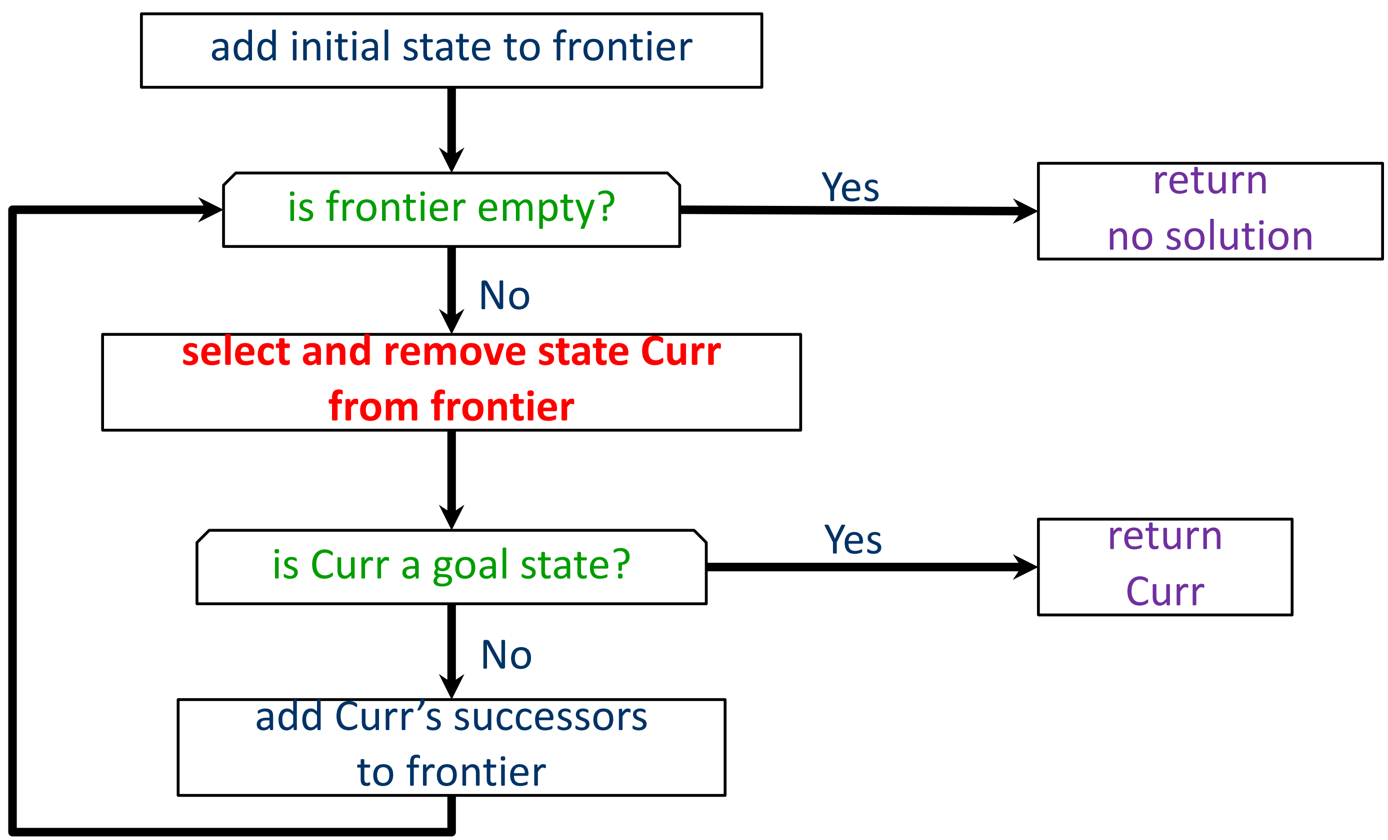
```
Search (Initial State, Successor Function, Goal Test)
    Frontier = { Initial State }
    While Frontier is not empty do
        Select and remove state Curr from Frontier
        If Curr is a goal state
            return Curr
        Add Curr’s Successors to Frontier
    Return no solution
```

Recovering the Path

How do we recover the path from the initial state if we return the goal node only?

Uninformed Search Strategy

The order in which we remove nodes from the frontier determines our search strategy and its properties.



Search Strategies

`Select and remove state Curr from Frontier`

Uninformed Search:

- DFS: remove newest state added
- BFS: remove oldest state added
- UCS: remove state with smallest total path cost

Heuristic Search:

- GBFS: remove state with smallest heuristic value
- A*: remove state with smallest (path cost + heuristic).

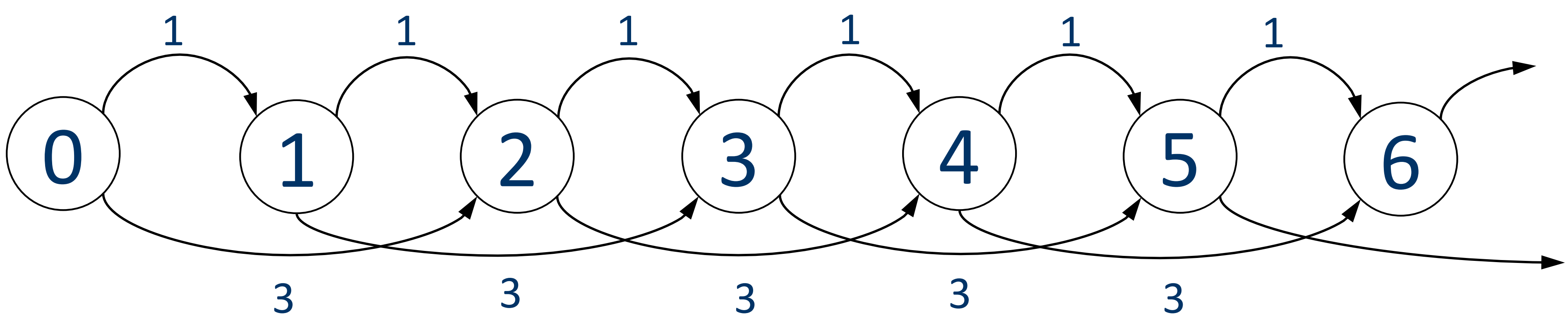
DEPTH-FIRST SEARCH

Depth-First Search

- The frontier is a stack (LIFO).
- Remove the most recent state added to the frontier.

Integer Example

- States: the non-negative integers $\{0, 1, 2, \ldots\}$.
- Initial state: 0.
- Goal state: 5.
- Successor function: $S(n) = \{n+1, n+2\}$.
- Cost function: $C(n, n+1) = 1, C(n, n+2) = 3$.



DFS on Integer Example

frontier:

search tree

Outcome of executing DFS on Integer example

What is the outcome of executing DFS on Integer example?

A. DFS finds a solution.
B. DFS finds the solution with the smallest total cost.
C. DFS does not terminate.

Properties of Search Algorithms

Completeness:

Will the search always find a solution if a solution exists?

Optimality:

Will the search always find the least-cost solution?

Time complexity:

What is the maximum number of states that we must visit?

Space complexity:

What is the maximum number of states that we must store in memory?

Useful Quantities

$$b$$

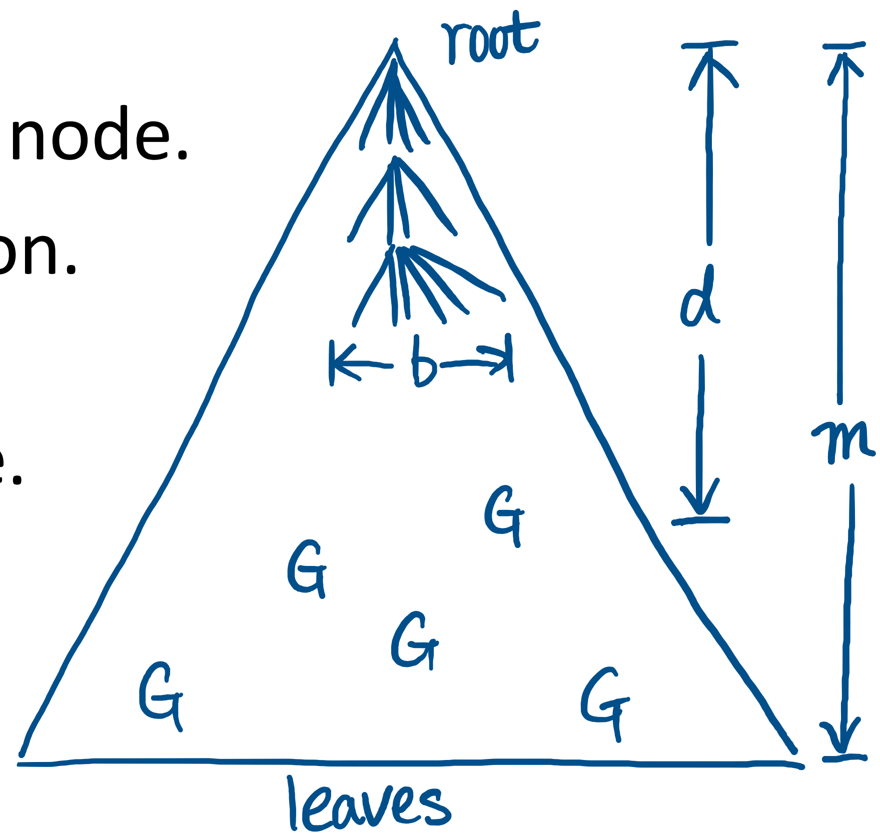
- Branching factor.
- Maximum number of successors of any state.

$$d$$

- Depth of the shallowest goal node.
- Length of the shortest solution.

$$m$$

- Max depth of the search tree.
- Length of the longest path.



DFS Properties – Completeness

Is DFS guaranteed to find a solution if a solution exists?

- No.
- Gets stuck in an infinite path.
  - E.g., integer example.
  - E.g., the search graph has a cycle.

DFS Properties – Optimality

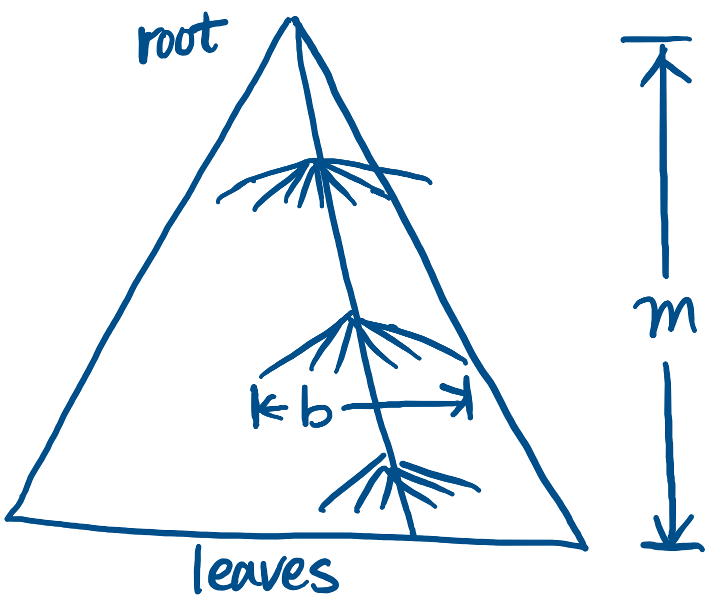
Is DFS guaranteed to find an optimal solution?

- No.
- Not optimizing costs.

DFS Properties – Space Complexity

$$O(bm)$$

- Linear in m.
- Explores a single path at a time.
- Remembers at most m nodes on the current path and at most b siblings for each node.



DFS Properties – Time Complexity

$$O(b^m)$$

- Exponential in m.
- Visits the entire search tree in the worst case.

Summary of Uninformed Search Algorithms

	DFS	BFS	IDS	UCS
Strategy	Remove newest state added			
Complete?	No			
Optimal?	No			
Space complexity	Linear in m $O(bm)$			
Time complexity	Exponential in m $O(b^m)$			

Question

What are the advantages of DFS?

A. It is complete.
B. It is optimal.
C. It has great space complexity.
D. It has great time complexity.

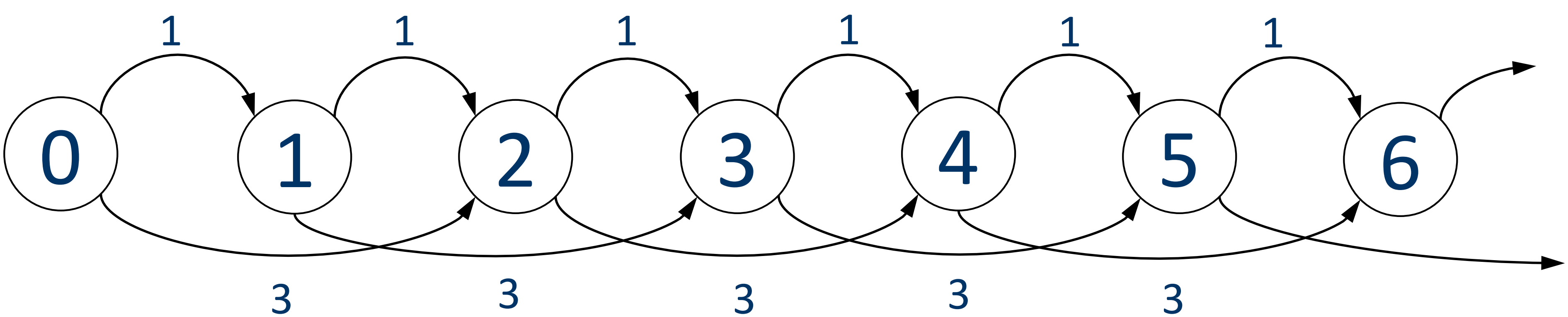
BREADTH-FIRST SEARCH

Breadth-First Search

- The frontier is queue (FIFO).
- Remove the oldest state added to the frontier.

Integer Example

- States: the non-negative integers $\{0, 1, 2, \ldots\}$.
- Initial state: 0.
- Goal state: 5.
- Successor function: $S(n) = \{n + 1, n + 2\}$.
- Cost function: $C(n, n + 1) = 1, C(n, n + 2) = 3$.



BFS on Integer Example

frontier:

search

tree

BFS Properties – Completeness

Is BFS guaranteed to find a solution if a solution exists?

- Yes.
- Guaranteed to find shallowest goal node at depth d.

BFS Properties – Optimality

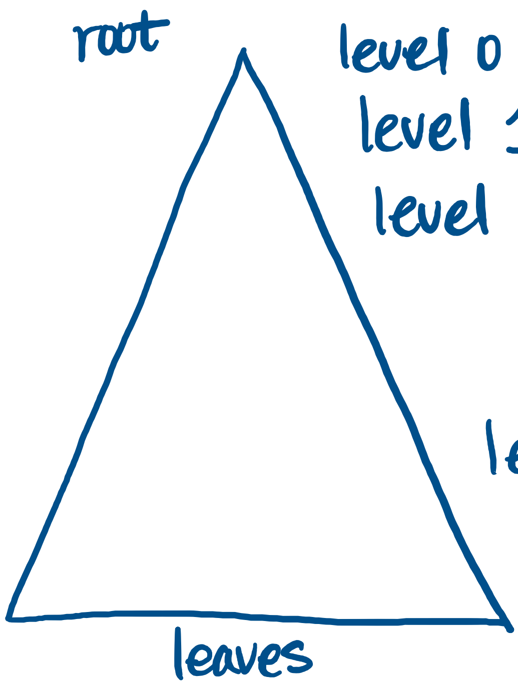
Is BFS guaranteed to find an optimal solution?

- No.
- Not optimizing costs.
- Guaranteed to find shallowest goal node at depth d.

BFS Properties – Time Complexity

$$O(b^d)$$

- Exponential in d.
- Must visit the top d levels.
- Frontier contains at most b^d nodes (on level d).



level 0 $\quad$ 1 node
level 1 $\quad \leq b$ nodes
level 2 $\quad \leq b^2$ nodes
$\vdots$
level d $\quad \leq b^d$ nodes

Total # of nodes explored:

$$1 + b + b^2 + b^3 + \cdots + b^d$$
$$= O(b^d)$$

BFS Properties – Space Complexity

$$O(b^d)$$

- Exponential in d.
- Visit the top d levels in the worst case.
- Total # of nodes dominated by # of nodes on level d.

Summary of Uninformed Search Algorithms

	DFS	BFS	IDS	UCS
Strategy	Remove newest state added	Remove oldest state added		
Complete?	No	Yes		
Optimal?	No	No		
Space complexity	Linear in m $O(bm)$	Exponential in d $O(b^d)$		
Time complexity	Exponential in m $O(b^m)$	Exponential in d $O(b^d)$		

Question 1: DFS v.s. BFS

Which search algorithm would you use
if the <u>search graph has cycles</u>?

A. I would use DFS but not BFS.
B. I would use BFS but no DFS.
C. Both DFS and BFS are appropriate.
D. Neither DFS nor BFS is appropriate.

Question 2: DFS v.s. BFS

Which search algorithm would you use
if the branching factor is infinite?

A. I would use DFS but not BFS.
B. I would use BFS but no DFS.
C. Both DFS and BFS are appropriate.
D. Neither DFS nor BFS is appropriate.

ITERATIVE-DEEPENING SEARCH

DFS versus BFS

Can we have a search algorithm that combines the best of DFS and BFS?

	Depth-First Search	**Breadth-First Search**
Space complexity	Linear in m $O(bm)$ 😄	Exponential in d $O(b^d)$
Completeness	May get stuck on an infinite path and never terminate.	Guaranteed to find a solution if it exists. 😄

Depth-Limited Search

Problem:

- DFS may follow an infinite path forever.

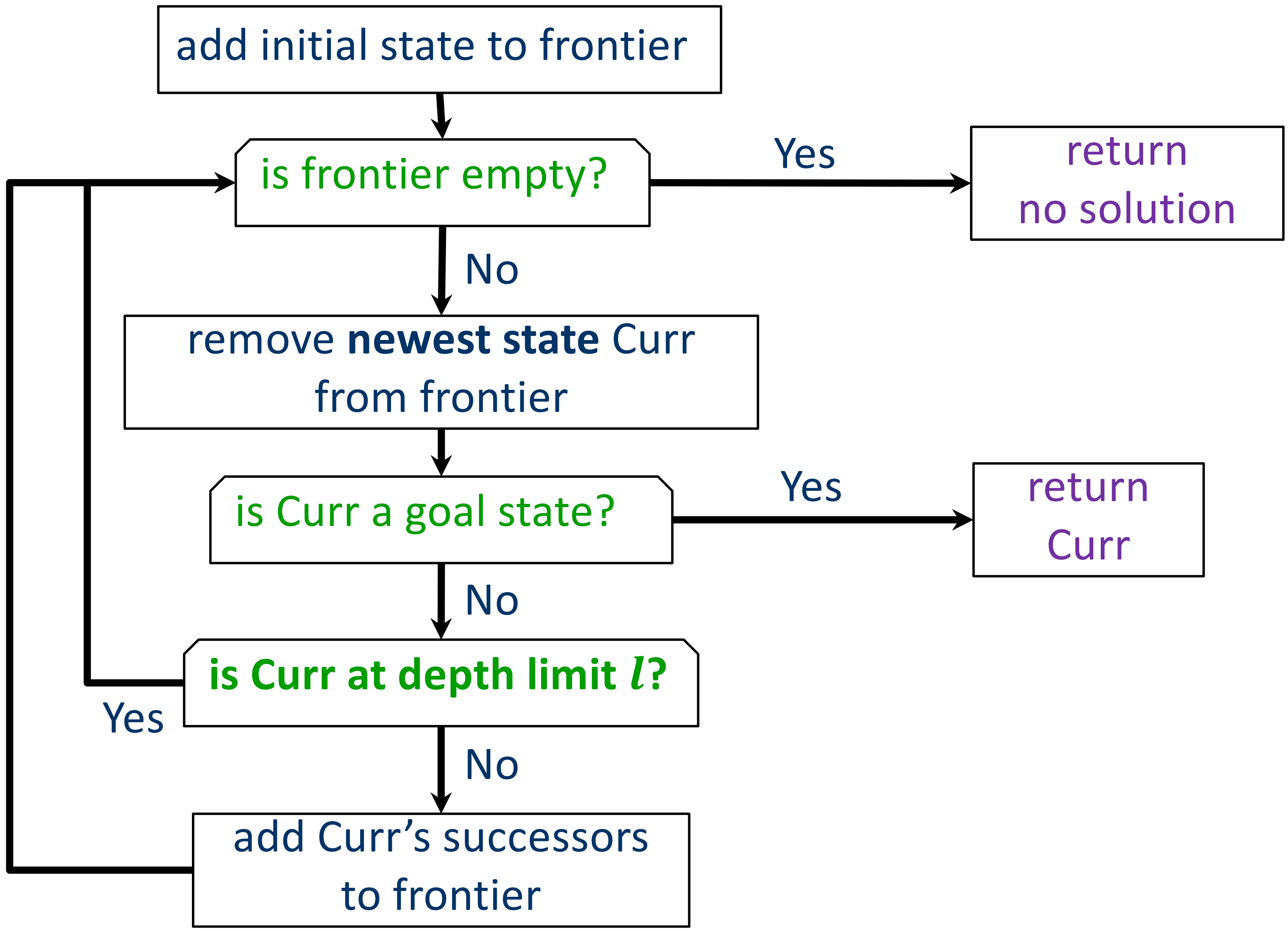
Idea:

- BFS is complete since it explores the tree level by level.

Solution:

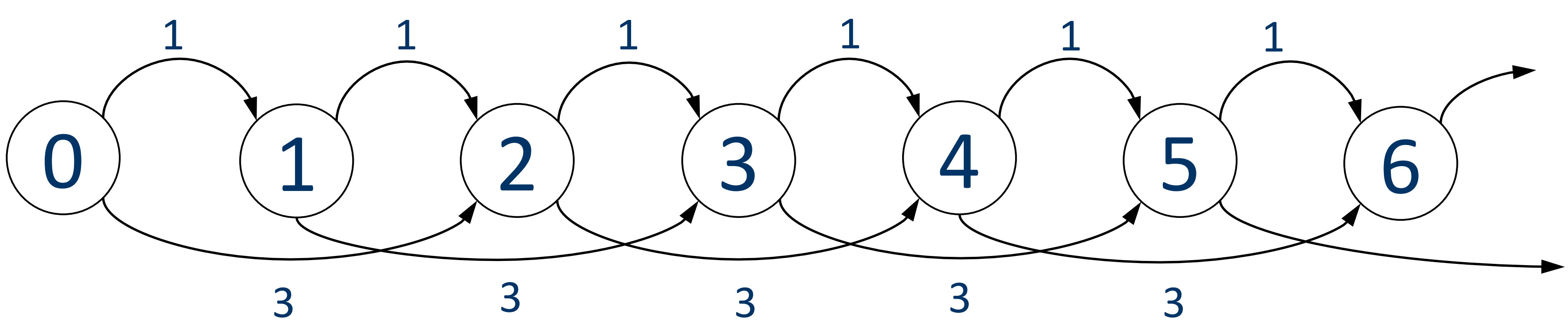
- Perform DFS up to a pre-specified depth limit l.

Depth-Limited Search as Flowchart



Integer Example

- States: the non-negative integers $\{0, 1, 2, \ldots\}$.
- Initial state: 0.
- Goal state: 5.
- Successor function: $S(n) = \{n + 1, n + 2\}$.
- Cost function: $C(n, n + 1) = 1$, $C(n, n + 2) = 3$.



DLS on Integer Example

Limit = 2

frontier:

search

tree

Depth-Limited Search

Desirable property:

- DLS no longer gets stuck on an infinite path.

Problem:

- If no goal node exists within the depth limit, DLS doesn't find a solution...

Solution: (If we do not succeed, try, try, try again...)

- If DLS doesn't find a solution w/ depth limit l, try again with depth limit $l + 1$.

Iterative-Deepening Search

- For depth limit 0 to ∞, perform depth-limited search.
- Depth-limited search:
    - When the node is goal, return the solution.
    - When depth reaches the limit, return without generating and adding successors.
    - Otherwise, add successors and continue the search.

IDS on Integer Example

limit =

frontier:

search

tree(s)

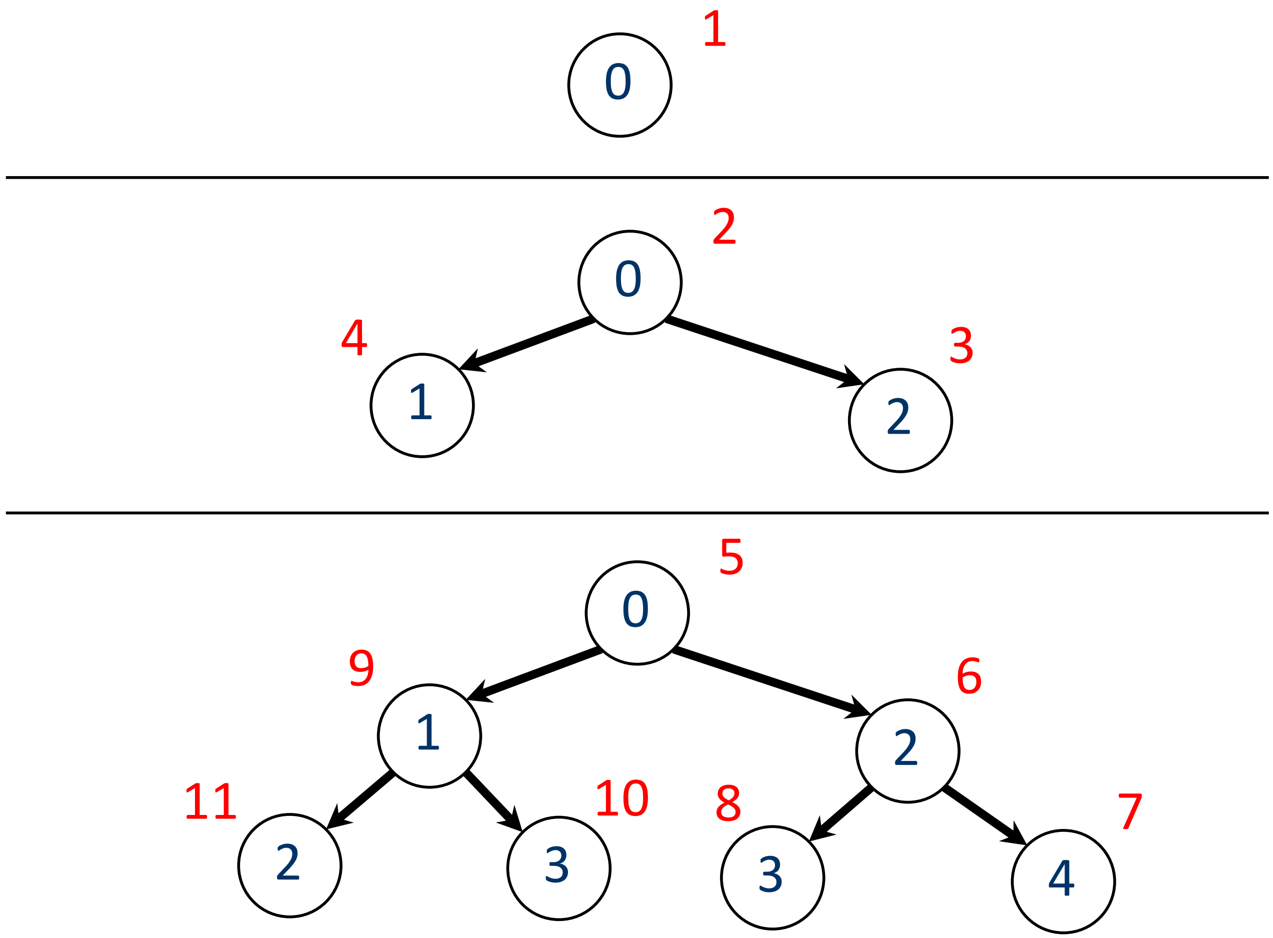
IDS on Integer Example

limit = ~~0~~, ~~1~~, ~~2~~

frontier: ~~0~~, ~~0~~, ~~1~~, ~~2~~, ~~0~~, ~~1~~, ~~2~~, ~~3~~, ~~4~~, ~~2~~, ~~3~~

search

tree(s)



Why restarting DLS from scratch?

Every time the depth limit l increases,

we start Depth-Limited Search from scratch $(l = 0)$.

When performing DLS for depth limit l,

can we reuse the work done for depth limit $l - 1$?

IDS Properties – Completeness

Is IDS guaranteed to find a solution if a solution exists?

A. Yes, IDS is complete.
B. No, IDS is not complete.

IDS Properties – Optimality

Is IDS guaranteed to find an optimal solution?

- No.
- Not optimizing costs.
- Guaranteed to find shallowest goal node ($\approx$ BFS).

IDS Properties – Space Complexity

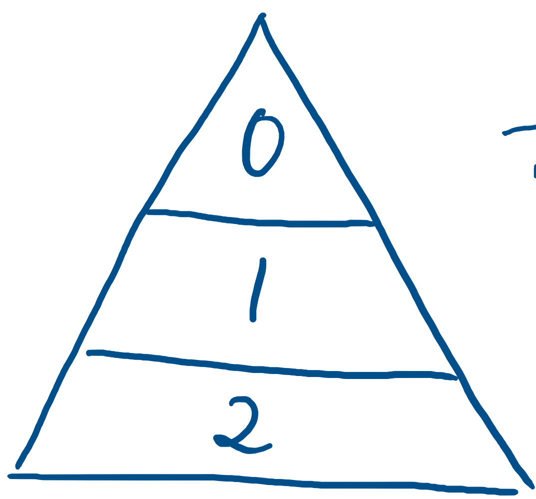
What is the space complexity for IDS?

A. Linear in d.
B. Linear in m.
C. Exponential in d.
D. Exponential in m.

IDS Properties – Time Complexity

$$O(b^d)$$

- Exponential in d.
- What about the repeated computations?
  - States in upper levels are generated multiple times.
  - Wasteful? Most of the states are in the bottom level.



depth limit	0	1	2
visits level 0	✓	✓	✓
level 1		✓	✓
level 2			✓

Summary of Uninformed Search Algorithms

	DFS	BFS	IDS	UCS
Strategy	Remove newest state added	Remove oldest state added	Remove newest state added	
Complete?	No	Yes	Yes	
Optimal?	No	No	No	
Space complexity	Linear in m $O(bm)$	Exponential in d $O(b^d)$	Linear in d $O(bd)$	
Time complexity	Exponential in m $O(b^m)$	Exponential in d $O(b^d)$	Exponential in d $O(b^d)$	

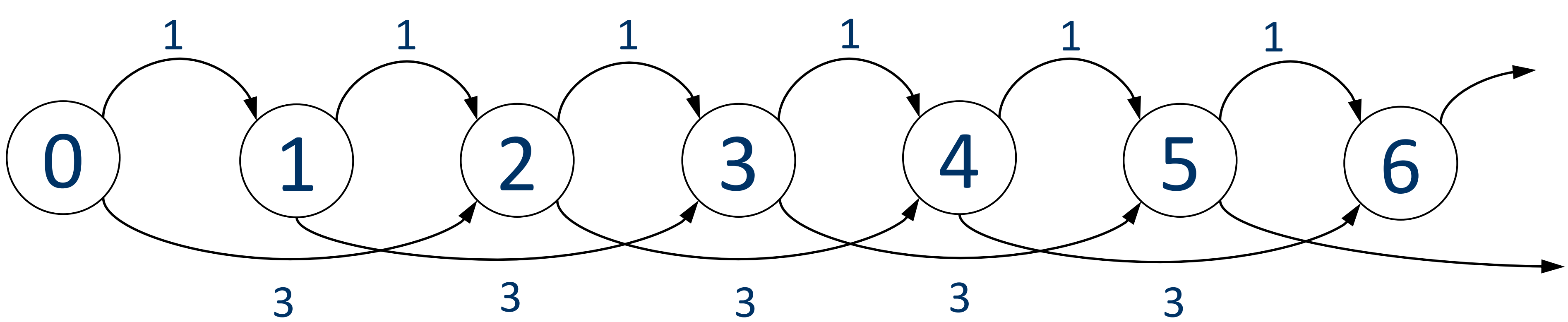
UNIFORM-COST SEARCH

Uniform-Cost Search

- The frontier is a priority queue ordered by path cost.
- Remove the least-cost path in the frontier.
- Also known as Dijkstra's shortest path algorithm.

Integer Example

- States: the non-negative integers $\{0, 1, 2, \ldots\}$.
- Initial state: 0.
- Goal state: 5.
- Successor function: $S(n) = \{n + 1, n + 2\}$.
- Cost function: $C(n, n + 1) = 1, C(n, n + 2) = 3$.



UCS on Integer Example

frontier:

search

tree

UCS Properties – Completeness and Optimality

Is UCS guaranteed to find a solution if a solution exists?

Is UCS guaranteed to find an optimal solution?

- Yes, under mild conditions.

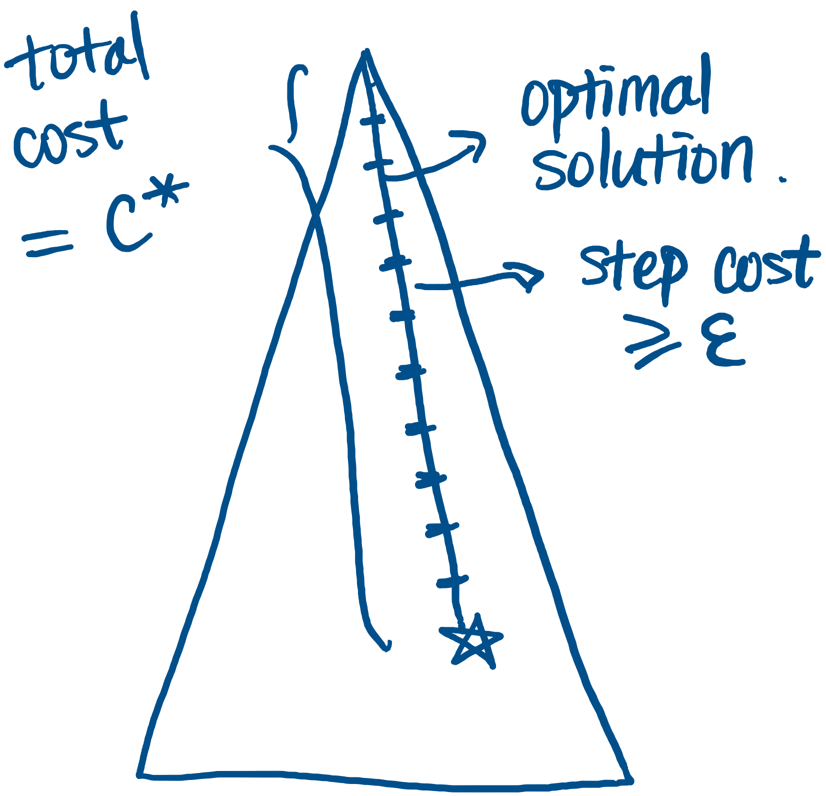
Mild conditions:

- The branching factor b is finite.
- The cost of every action is bounded below by a positive constant ($\varepsilon > 0$).

UCS Properties – Time and Space Complexity

$$O\left(b^{\frac{C^*}{\epsilon}}\right)$$

- C^* is the cost of the optimal solution.
- ε is the minimal cost of an action.



1. total cost
2. cost of each step
3. # of steps
4. level of optimal solution $\leq C^*/\varepsilon$.

Summary of Uninformed Search Algorithms

	DFS	BFS	IDS	UCS
Strategy	Remove newest state added	Remove oldest state added	Remove newest state added	Remove state with smallest path cost
Complete?	No	Yes	Yes	Yes
Optimal?	No	No	No	Yes
Space complexity	Linear in m $O(bm)$	Exponential in d $O(b^d)$	Linear in d $O(bd)$	$O(b^{\frac{C^*}{e}})$
Time complexity	Exponential in m $O(b^m)$	Exponential in d $O(b^d)$	Exponential in d $O(b^d)$	$O(b^{\frac{C^*}{e}})$

Question

If each edge has the same cost, UCS and BFS are identical.

True or False

MULTIPLE-PATH PRUNING

What is Multiple-Path Pruning?

Problem:

- There can be multiple paths to the same state
- We only need to find one (or the best) path to a state.

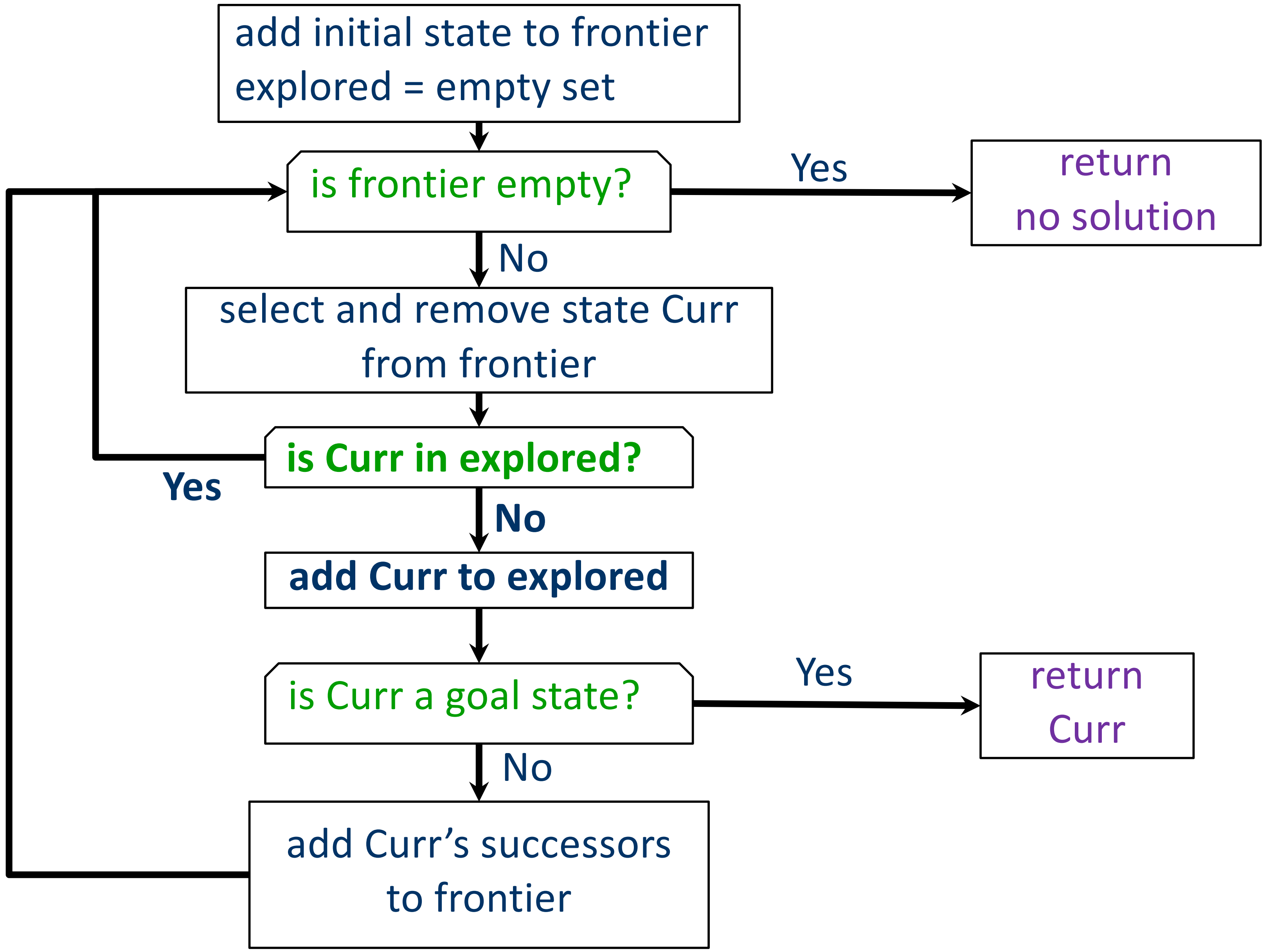
Solution:

- Keep only one path to each state.

Key ideas:

- Remembers all the expanded states in a set.
- Expand a node only if it's not in the set.

Generic Search Algorithm with Pruning



Generic Search Algorithm **with Pruning**

```
Search (Initial State, Successor Function, Goal Test)
  Frontier = { Initial State }
  Explored = { }
  While Frontier is not empty do
     Select and remove state Curr from Frontier
     If Curr is NOT in Explored
        Add Curr to Explored
        If Curr is a goal state
           return Curr
        Add Curr’s Successors to Frontier
  Return no solution
```

BFS with Pruning on Integer Example

frontier:

explored:

search

tree

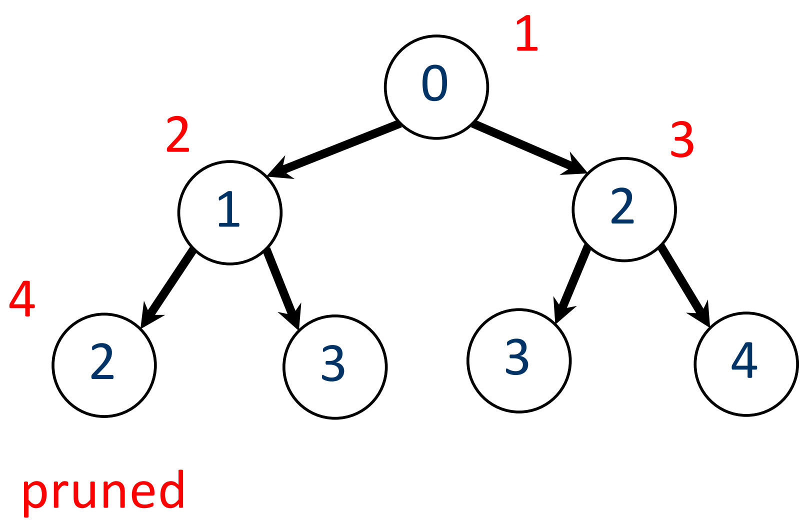
BFS with Pruning on Integer Example

frontier: 0, 01, 02, 012, 013, 023, 024

explored: 0, 1, 2

search

tree



Effects on Space Complexity

- Multiple-path pruning incurs high space complexity.
- Combining with BFS?
- Combining with DFS?

Question

How does Multi-Path Pruning
affect the space complexity of DFS?

A. No effect
B. Pruning changes the space complexity of DFS from linear to exponential.
C. Pruning changes the space complexity of DFS from exponential to linear.

Effects on Optimality

Will UCS with multi-path pruning

find an optimal solution?

- Yes.
- The first path to each state found by UCS must be the lowest-cost path to the state.
- The pruned paths cannot have lower costs than the first path UCS found.

This no longer holds for some heuristic search algorithms.