SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

# Uninformed Search II

*PROF. LIM KWAN HUI*

50.021 Artificial Intelligence

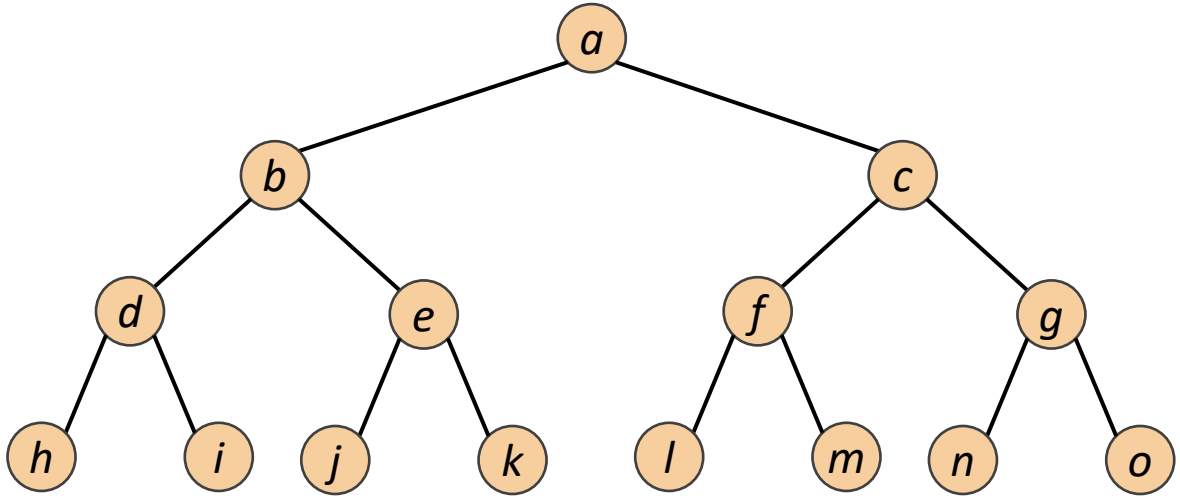
*The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.*

# Types of Uninformed Search

- ~~Breadth-First Search~~
- ~~Uniform-cost search~~
- Depth-First Search
- Depth-limited search
- Iterative deepening search

# Depth-First Search

- General idea: Expand the *deepest* unexpanded node
- Implementation: Using a *Last-In First-Out (LIFO)* queue

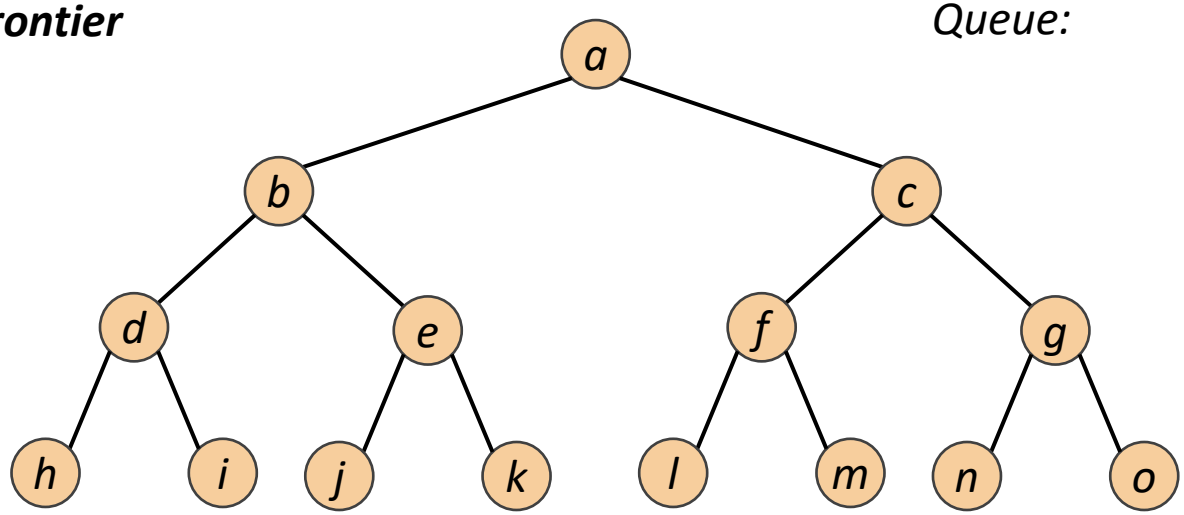


# Depth-First Search

- General idea: Expand the *deepest* unexpanded node
- Implementation: Using a *Last-In First-Out (LIFO)* queue

***Initialize frontier***

*Queue:*

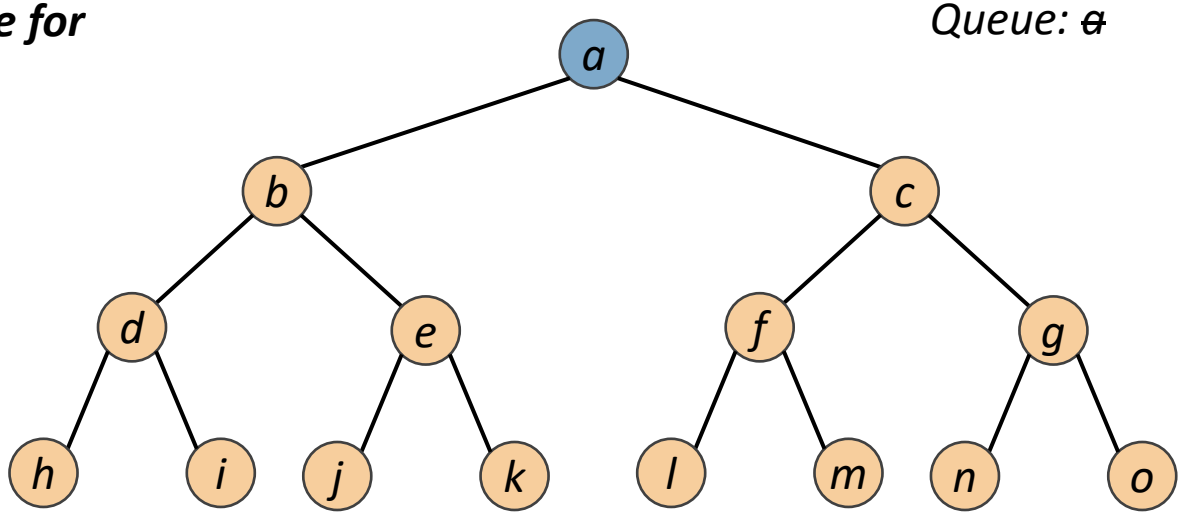


# Depth-First Search

- General idea: Expand the *deepest* unexpanded node
- Implementation: Using a *Last-In First-Out (LIFO)* queue

***Check node for goal state***

*Queue: ~~a~~*



# Depth-First Search

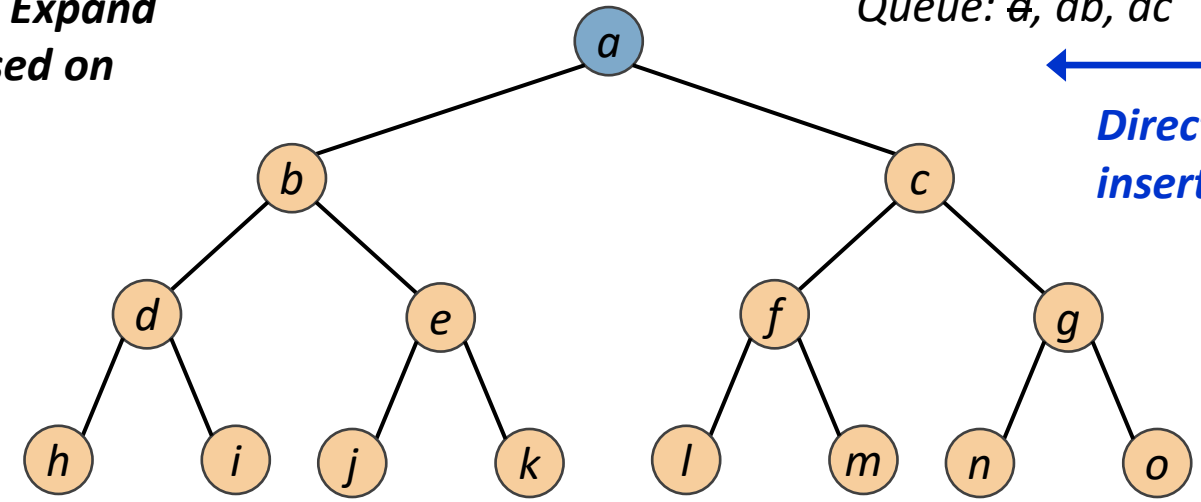
- General idea: Expand the *deepest* unexpanded node
- Implementation: Using a *Last-In First-Out (LIFO)* queue

***Insertion order: Insert by lowest alphabetical order first***

***Not goal? Expand nodes based on strategy***

*Queue: ~~a~~, ab, ac*

***Direction of queue insertion***



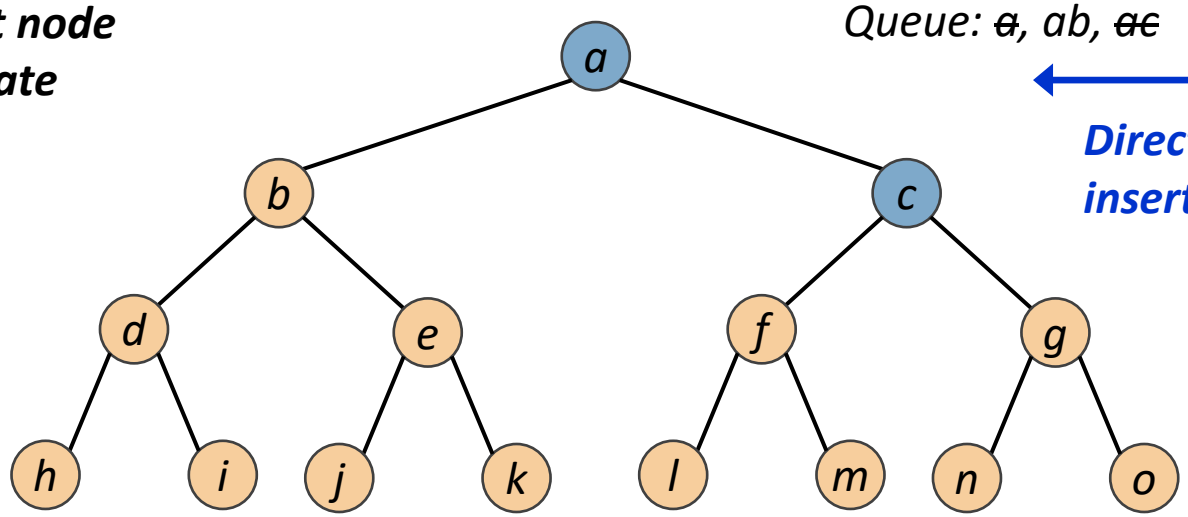
# Depth-First Search

- General idea: Expand the *deepest* unexpanded node
- Implementation: Using a *Last-In First-Out (LIFO)* queue

***Insertion order: Insert by lowest alphabetical order first***

***Check next node for goal state***

*Queue: ~~a~~, ab, ~~ac~~*




***Direction of queue insertion***

# Depth-First Search

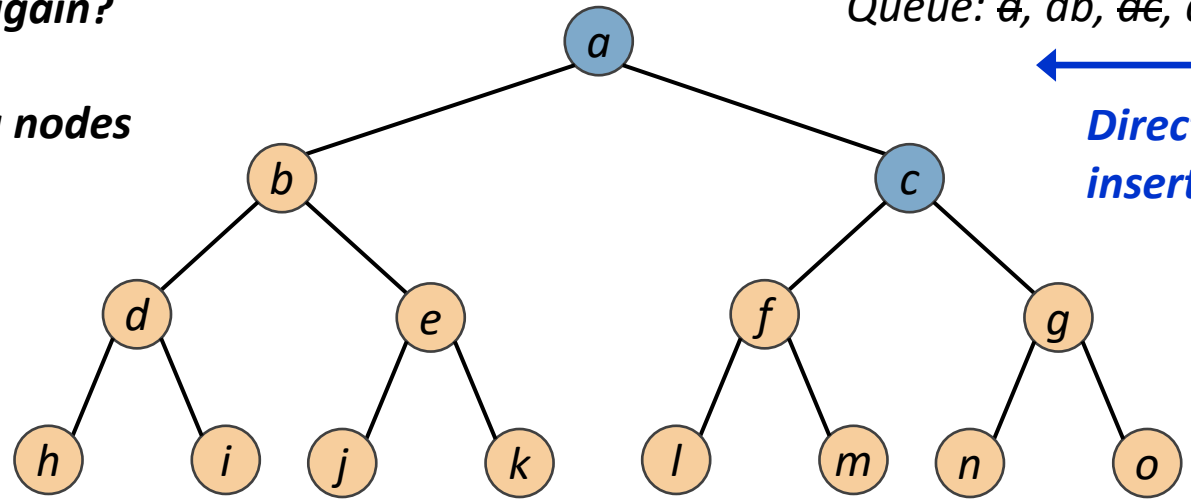
- General idea: Expand the *deepest* unexpanded node
- Implementation: Using a *Last-In First-Out (LIFO)* queue

***Insertion order: Insert by lowest alphabetical order first***

***Not goal again? Continue expanding nodes***

*Queue: ~~a~~, ab, ~~ac~~, acf, acg*

***Direction of queue insertion***



# Depth-First Search

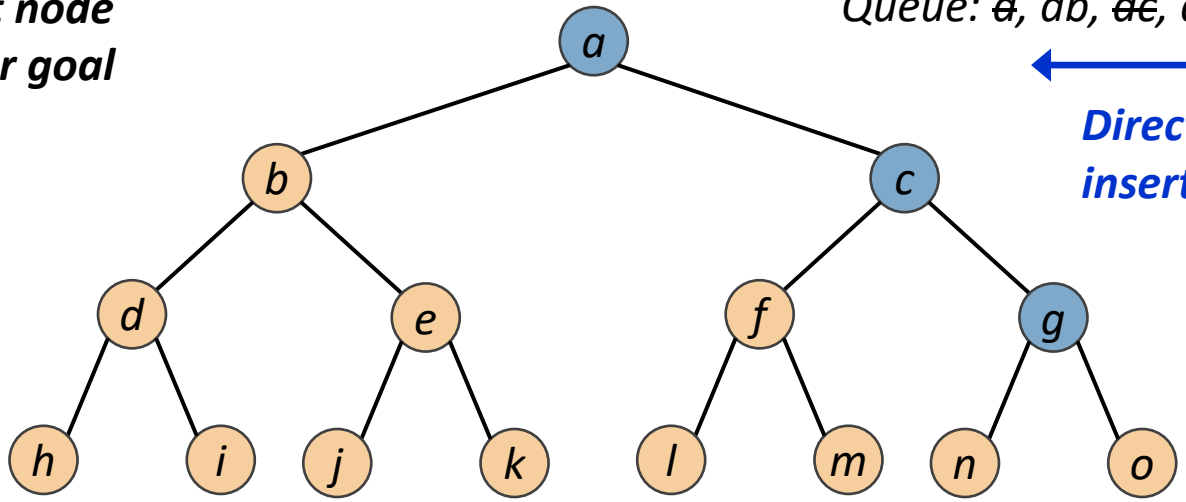
- General idea: Expand the *deepest* unexpanded node
- Implementation: Using a *Last-In First-Out (LIFO)* queue

***Insertion order: Insert by lowest alphabetical order first***

***Check next node (again!) for goal state***

*Queue: ~~a~~, ab, ~~ac~~, acf, ~~acg~~*

***Direction of queue insertion***



# Depth-First Search

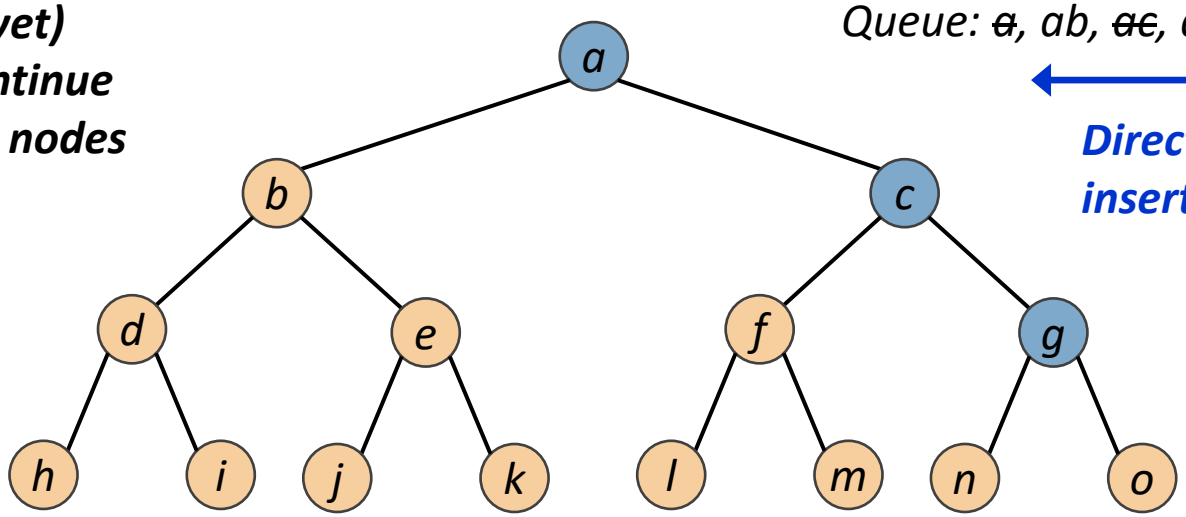
- General idea: Expand the *deepest* unexpanded node
- Implementation: Using a *Last-In First-Out (LIFO)* queue

***Insertion order: Insert by lowest alphabetical order first***

***Not goal (yet) again? Continue expanding nodes (again!)***

*Queue: ~~a~~, ab, ~~ac~~, acf, ~~acg~~, acgn, acgo*

***Direction of queue insertion***



# Depth-First Search

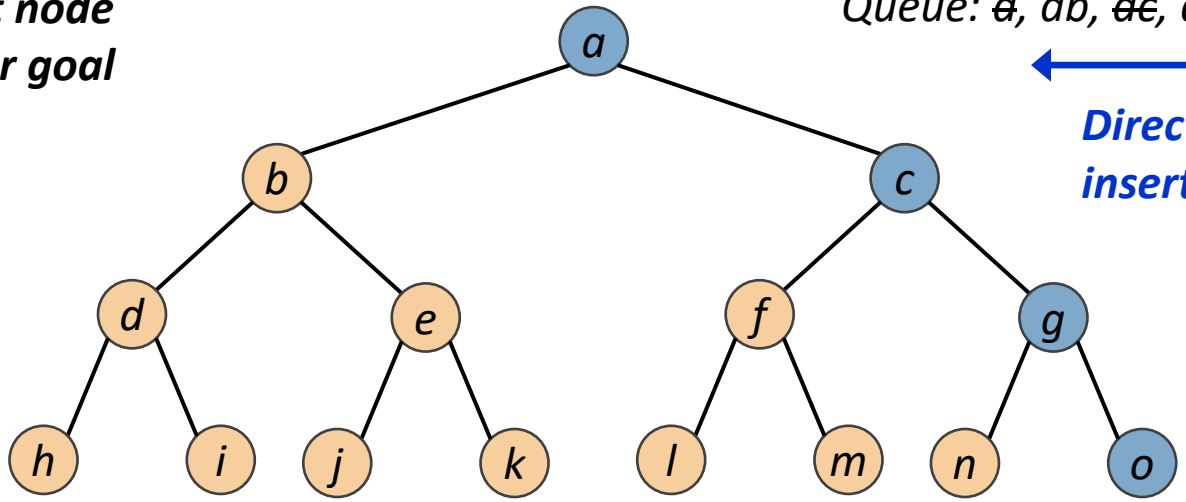
- General idea: Expand the *deepest* unexpanded node
- Implementation: Using a *Last-In First-Out (LIFO)* queue

***Insertion order: Insert by lowest alphabetical order first***

***Check next node (again!) for goal state***

*Queue: ~~a~~, ab, ~~ac~~, acf, ~~acg~~, acgn, ~~acgo~~*

***Direction of queue insertion***



# Depth-First Search

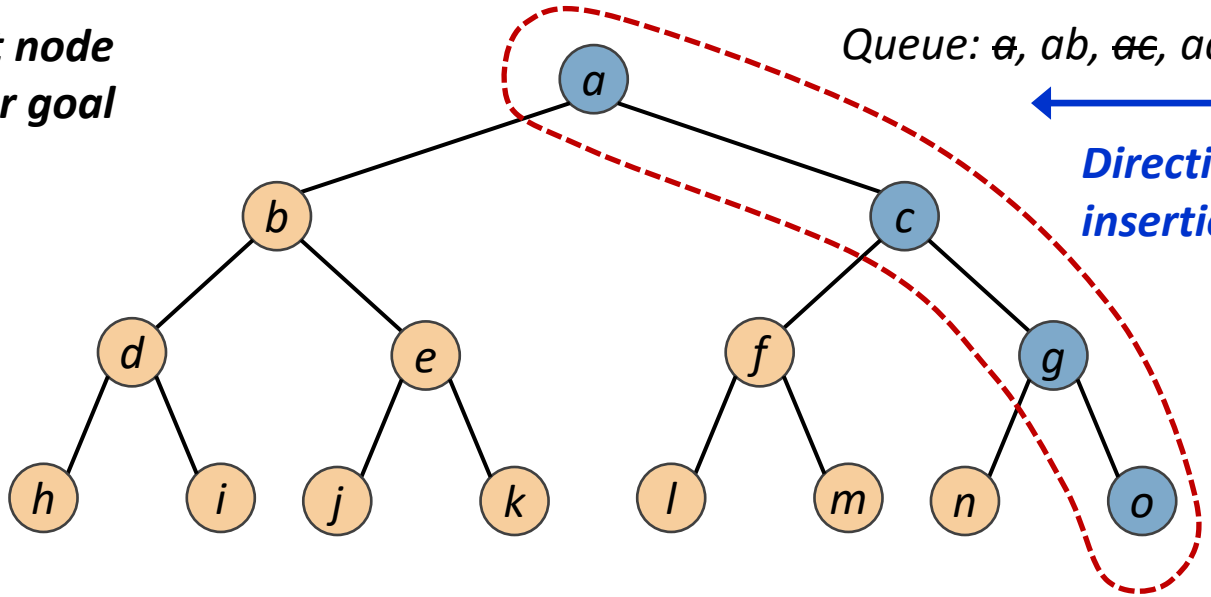
- General idea: Expand the *deepest* unexpanded node
- Implementation: Using a *Last-In First-Out (LIFO)* queue

***Insertion order: Insert by lowest alphabetical order first***

***Check next node (again!) for goal state***

*Queue: ~~a~~, ab, ~~ac~~, acf, ~~acg~~, acgn, ~~acgo~~*

***Direction of queue insertion***



# Properties of Depth-First Search

- Completeness: No

  (if *m* is infinite)
- Optimality: No
- Time complexity:
- Space complexity:

# Properties of Depth-First Search

- Completeness: No

  (if $m$ is infinite)
- Optimality: No
- Time complexity: $O(b^m)$

  (terrible if $m > d$ by a lot!)
- Space complexity:

# Properties of Depth-First Search

- Completeness: No
  (if $m$ is infinite)
- Optimality: No
- Time complexity: $O(b^m)$
  (terrible if $m > d$ by a lot!)
- Space complexity: $O(bm)$
  (linear space!)

# Exercise: Breadth-First Search VS Depth-First Search

- When is Breadth-First Search preferred over Depth-First Search?

- When is Depth-First Search preferred over Breadth-First Search?

# Exercise: Breadth-First Search VS Depth-First Search

- When is Breadth-First Search preferred over Depth-First Search?
  - When optimal solution is important
  - When *m* is much greater than *d* (as DFS takes $O(b^m)$ time)

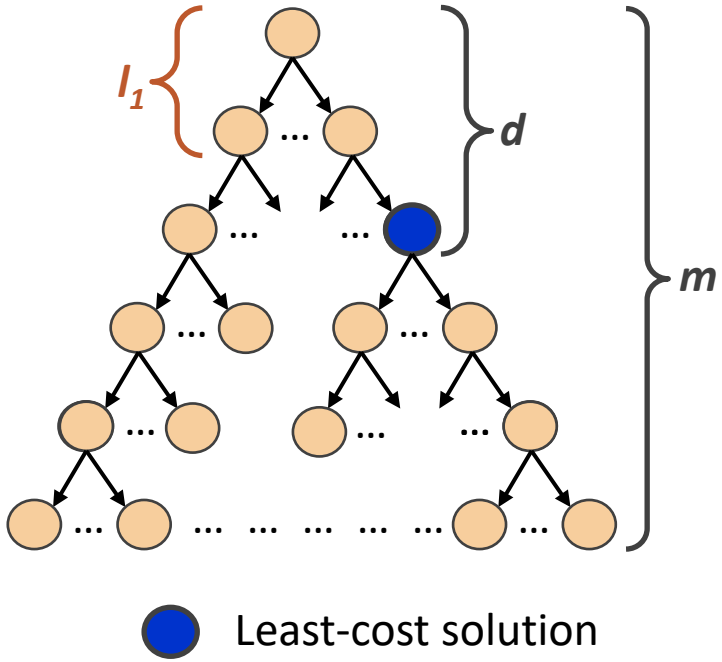
- When is Depth-First Search preferred over Breadth-First Search?
  - When space is important. DFS requires *O(bm), BFS requires* $O(b^d)$

# Depth-Limited Search

- General idea: Depth-First Search with predetermined depth limit *l*
  - i.e. nodes at depth *l* have no successors (child nodes)
  - Solves the infinite-path problem
- Completeness: No (if $l < d$)
- Optimality: No (if $l > d$)
- Time complexity: $O(b^l)$
- Space complexity: $O(bl)$

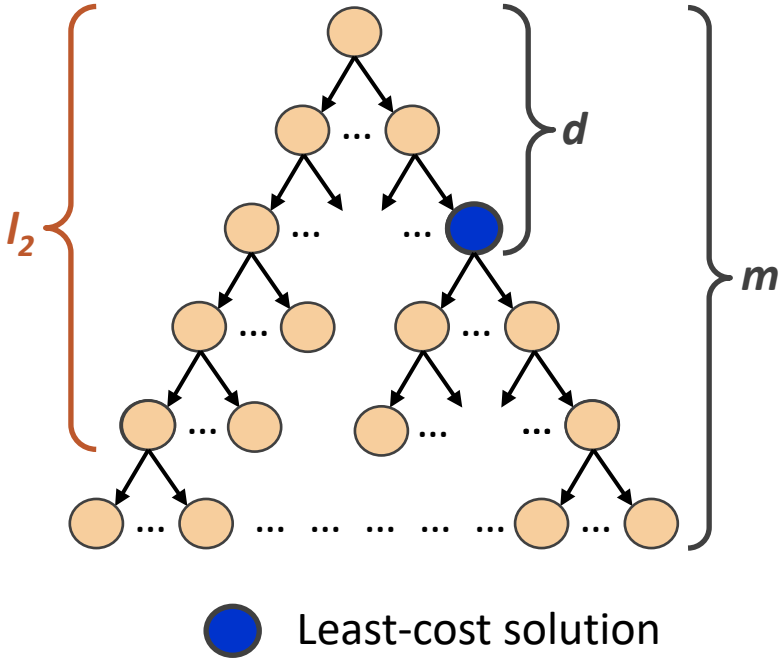
# Depth-Limited Search

- General idea: Depth-First Search with predetermined depth limit *l*
  - i.e. nodes at depth *l* have no successors (child nodes)
  - Solves the infinite-path problem
- Completeness: No (if $l < d$)
- Optimality: No (if $l > d$)
- Time complexity: $O(b^l)$
- Space complexity: $O(bl)$



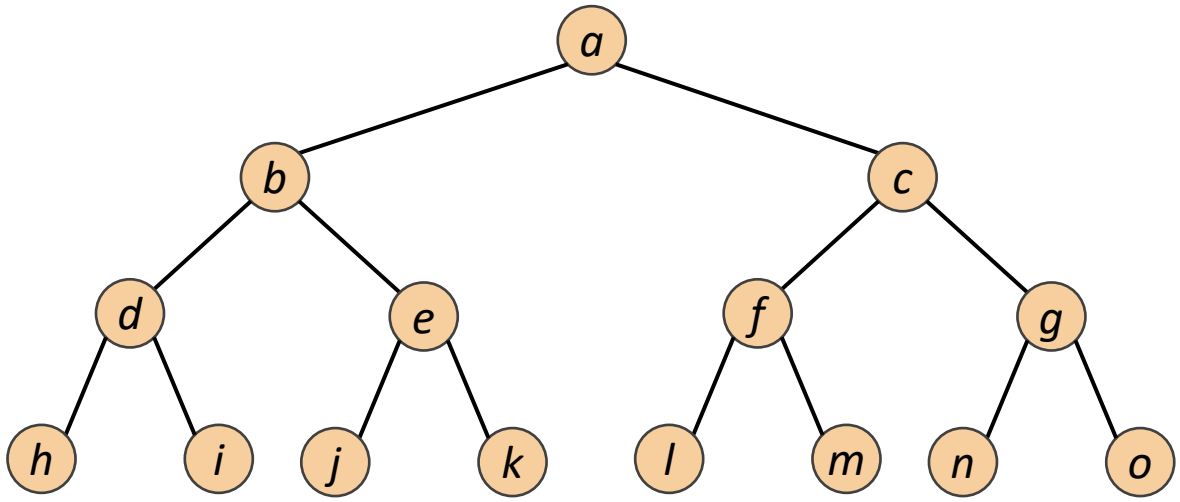
# Depth-Limited Search

- General idea: Depth-First Search with predetermined depth limit *l*
  - i.e. nodes at depth *l* have no successors (child nodes)
  - Solves the infinite-path problem
- Completeness: No (if $l < d$)
- Optimality: No (if $l > d$)
- Time complexity: $O(b^l)$
- Space complexity: $O(bl)$



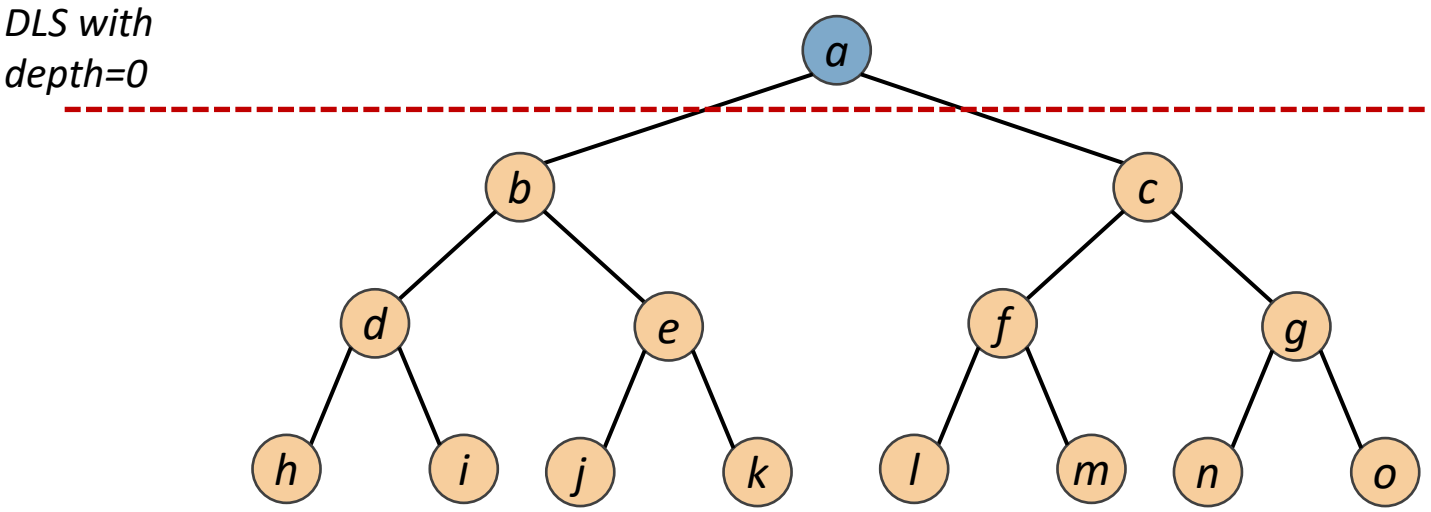
# Iterative Deepening Search

- General idea: Use increasing Depth-Limited Search (DLS) to find the best depth limit *l*
  - I.e., use DLS with depth limit 1. If no solution, then increase depth limit to 2. So on and so on, until solution is found
- Best of both Breadth-First Search and Depth-First Search



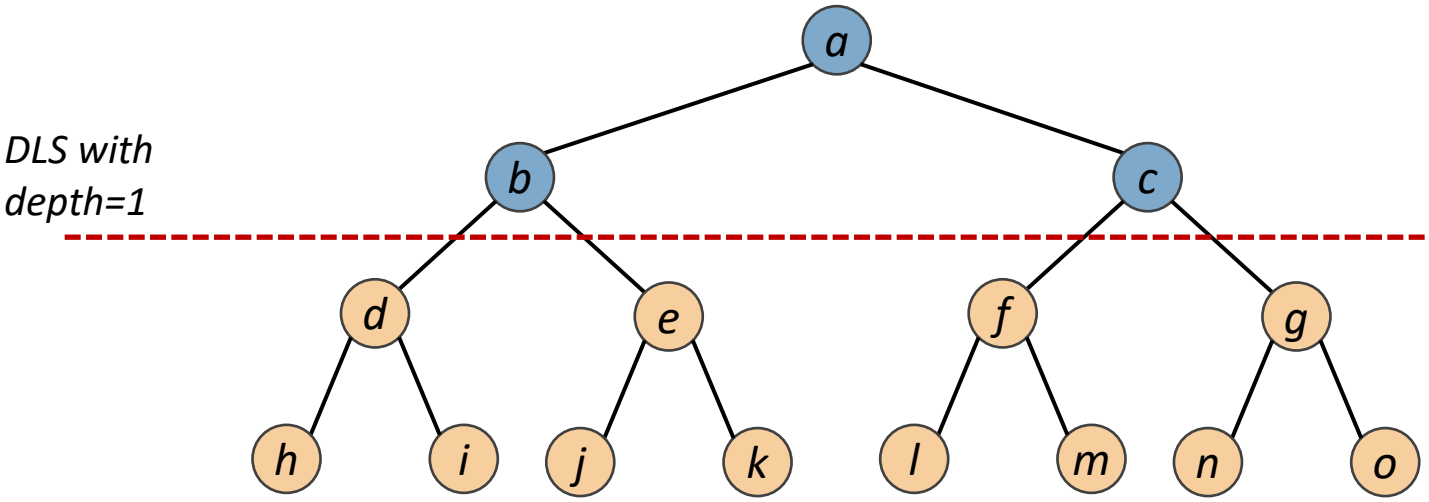
# Iterative Deepening Search

- General idea: Use increasing Depth-Limited Search (DLS) to find the best depth limit *l*
  - I.e., use DLS with depth limit 1. If no solution, then increase depth limit to 2. So on and so on, until solution is found
- Best of both Breadth-First Search and Depth-First Search



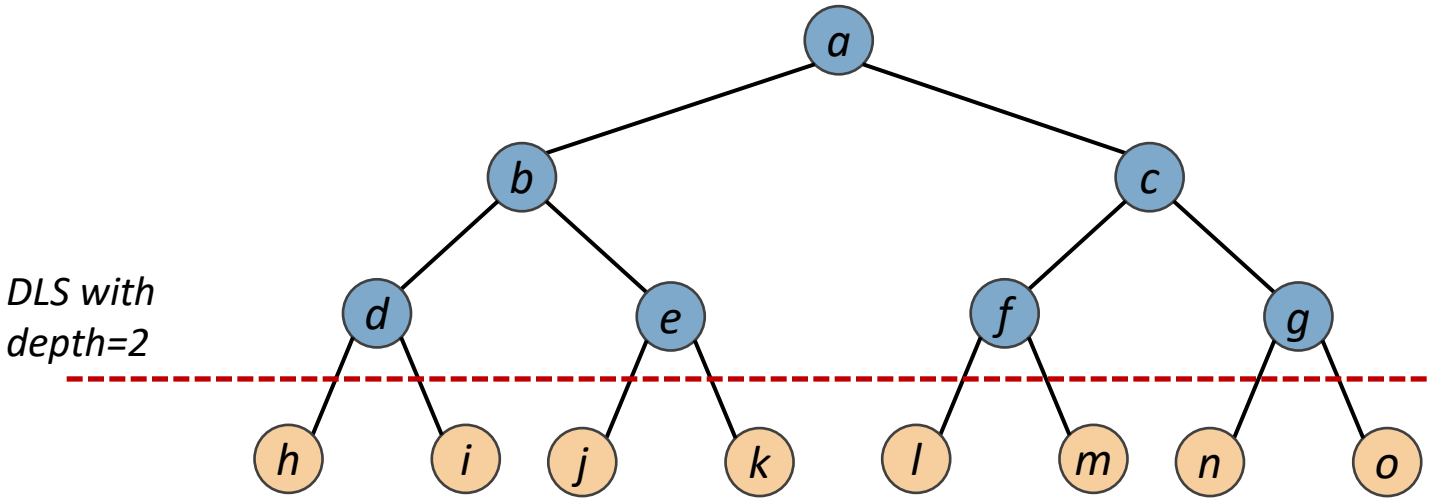
# Iterative Deepening Search

- General idea: Use increasing Depth-Limited Search (DLS) to find the best depth limit *l*
  - I.e., use DLS with depth limit 1. If no solution, then increase depth limit to 2. So on and so on, until solution is found
- Best of both Breadth-First Search and Depth-First Search



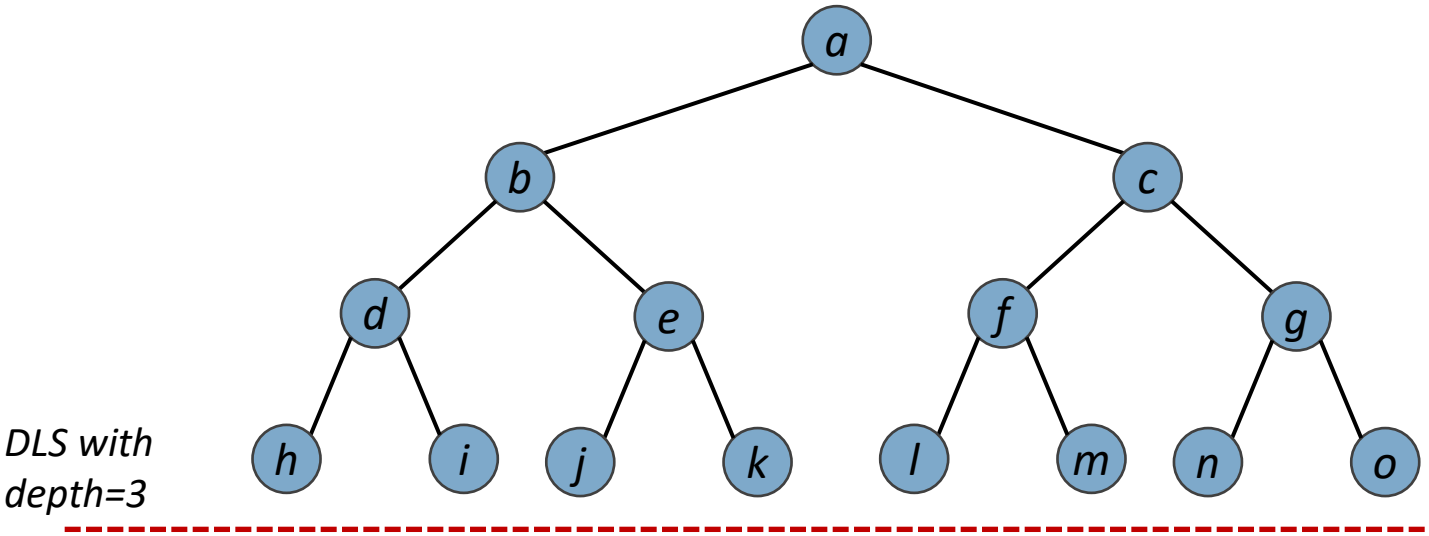
# Iterative Deepening Search

- General idea: Use increasing Depth-Limited Search (DLS) to find the best depth limit *l*
  - I.e., use DLS with depth limit 1. If no solution, then increase depth limit to 2. So on and so on, until solution is found
- Best of both Breadth-First Search and Depth-First Search



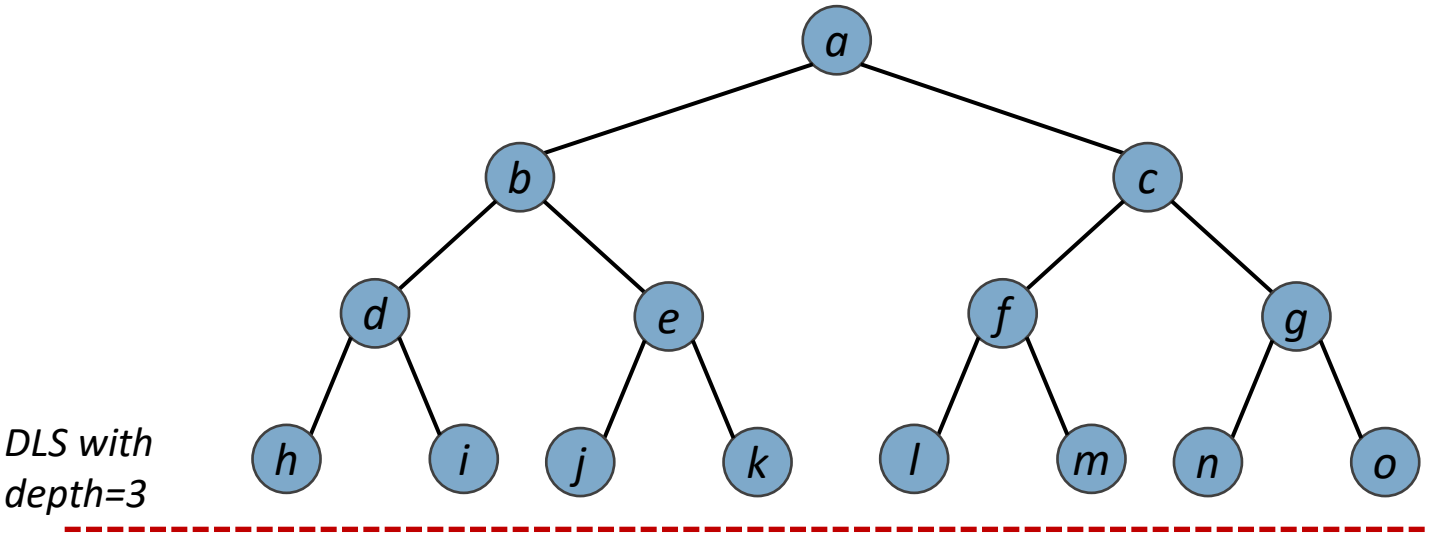
# Iterative Deepening Search

- General idea: Use increasing Depth-Limited Search (DLS) to find the best depth limit *l*
  - I.e., use DLS with depth limit 1. If no solution, then increase depth limit to 2. So on and so on, until solution is found
- Best of both Breadth-First Search and Depth-First Search



# Iterative Deepening Search

- General idea: Use increasing Depth-Limited Search (DLS) to find the best depth limit *l*
  - I.e., use DLS with depth limit 1. If no solution, then increase depth limit to 2. So on and so on, until solution is found
- Best of both Breadth-First Search and Depth-First Search



# Iterative Deepening Search

- General idea: Use increasing Depth-Limited Search (DLS) to find the best depth limit *l*
  - I.e., use DLS with depth limit 1. If no solution, then increase depth limit to 2. So on and so on, until solution is found
- Best of both Breadth-First Search and Depth-First Search
- But is it wasteful to keep generating states with each increasing DLS?

# Iterative Deepening Search

- General idea: Use increasing Depth-Limited Search (DLS) to find the best depth limit *l*
  - I.e., use DLS with depth limit 1. If no solution, then increase depth limit to 2. So on and so on, until solution is found
- Best of both Breadth-First Search and Depth-First Search
- But is it wasteful to keep generating states with each increasing DLS?
  - Turns out it is not too costly as most of the work (node expansion) happens at the lower depths

# Properties of Iterative Deepening Search

- Completeness: Yes
- Optimality: Yes
- Time complexity: $O(b^d)$
- Space complexity: $O(bd)$

# Summary: Uninformed Search

- Breadth-First Search
- Uniform-cost search
- Depth-First Search
- Depth-limited search
- Iterative deepening search