Lecture 12
Uninformed Search Strategies

Artificial Intelligence

Dr. Ahmed Mateen

Agenda

- Depth Limited Search
- Iterative Deepening Search

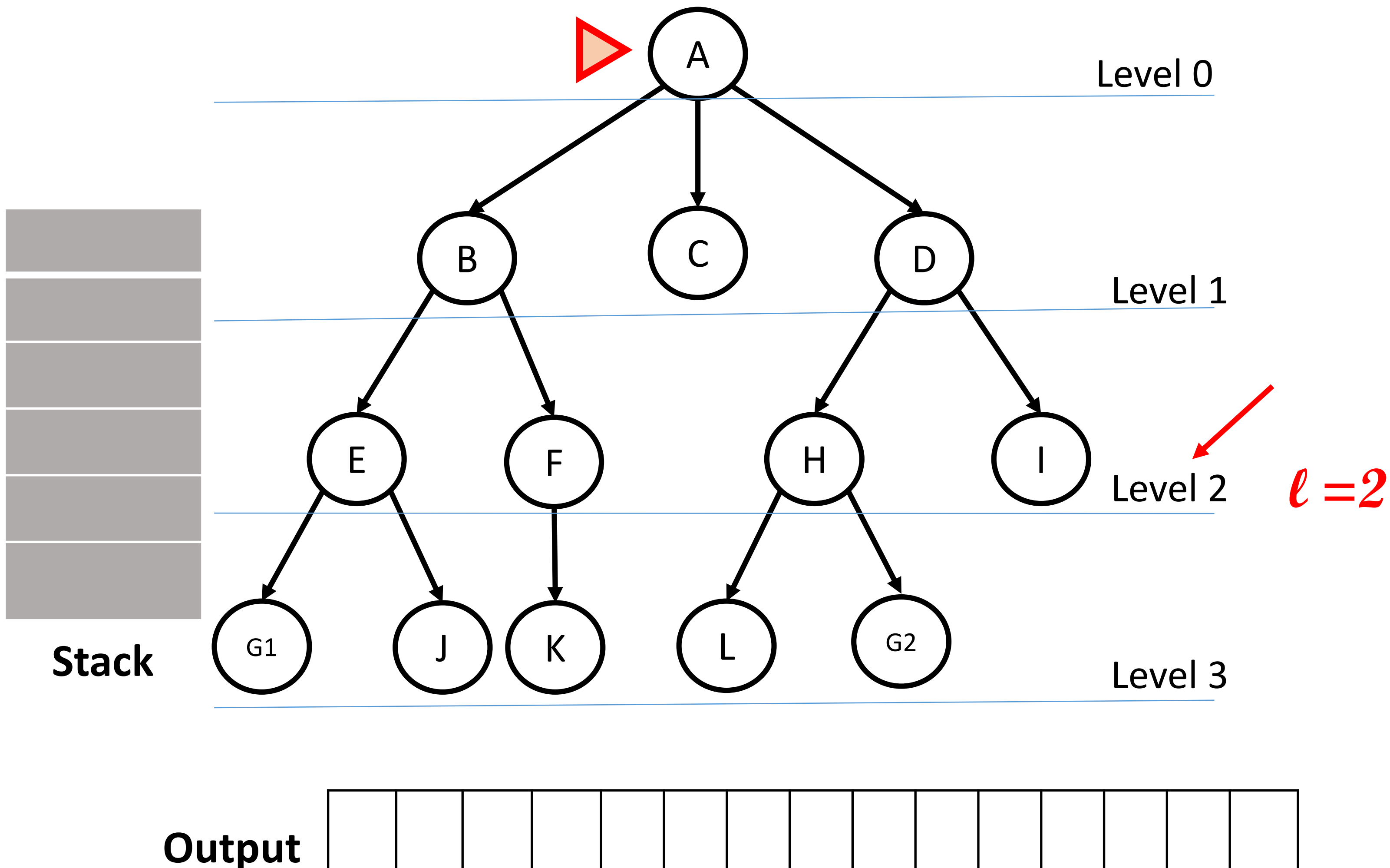
Depth Limited Search (DLS)

- Same as DFS with level limitation or depth limit.
  - Search is limited up to some predetermined level ℓ .
  - Nodes at depth ℓ have no successors.
  - Alleviates the problem of unbounded trees

DLS = DFS + Limit for the level

Depth Limited Search (DLS)

1. Select level
2. Apply DFS up-to the selected level



Depth Limited Search (DLS)

- **Limitation->** If goal node is located after the height limit, it will fail.
- **Benefit->** Will not go into infinite loops.

- Same as depth-first search if $\boldsymbol{\ell} = \infty$

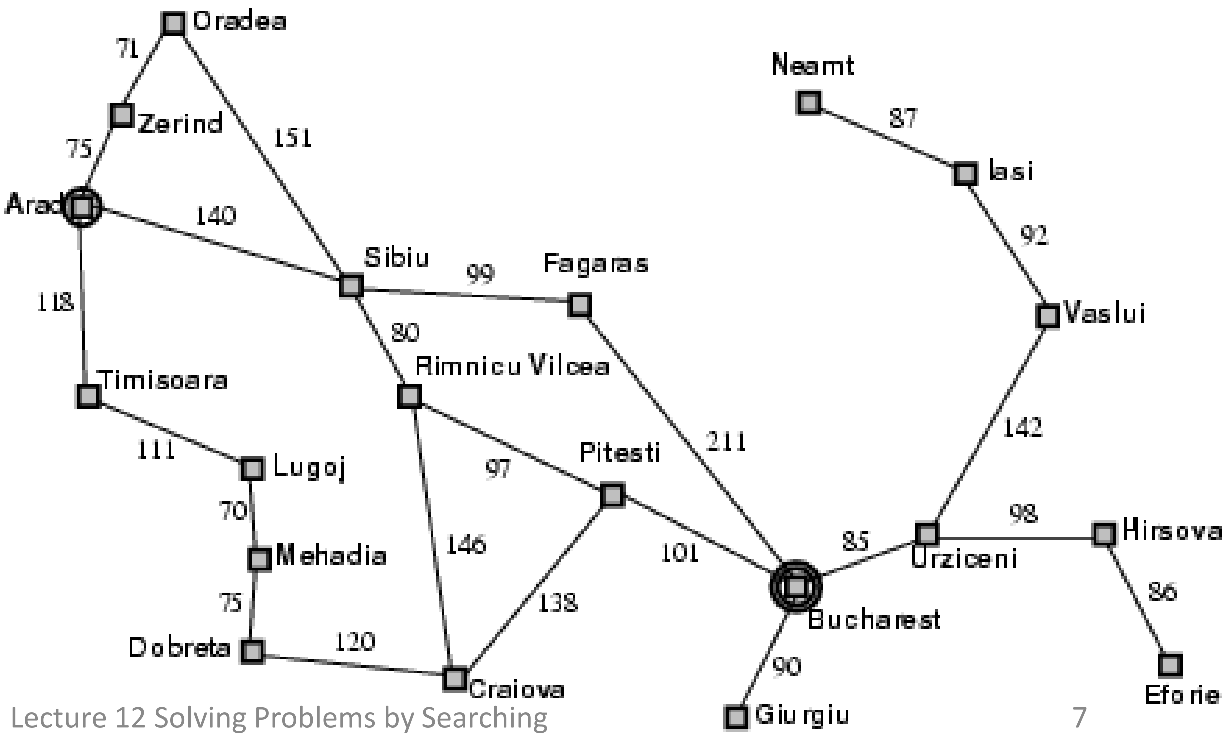
Depth-Limited Search

- Complete
  - DLS search algorithm is complete if the solution is above the depth-limit.
- Time
  - $O(b^{\ell})$
- Space
  - $O(b\boldsymbol{\ell})$
- Optimal
  - Not even if $\boldsymbol{\ell} > d$

Depth Limited Search

- How to choose *ℓ* ?
- Sometimes based on knowledge of the problem

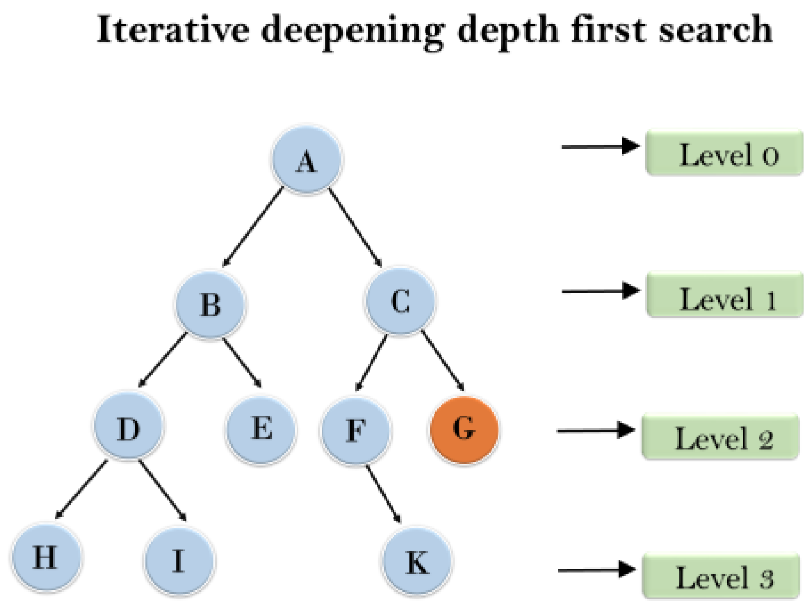
ℓ =*19*



Iterative Deepening Search (IDS)

- Iterative deepening depth-first search
  - Uses depth-first search
  - Finds the best depth limit
    - Gradually increases the depth limit; 0, 1, 2, … until a goal is found

Iterative Deepening Search



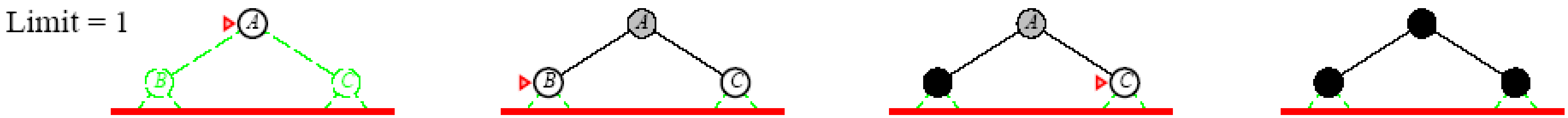
1'st Iteration-----> A

2'nd Iteration----> A, B, C

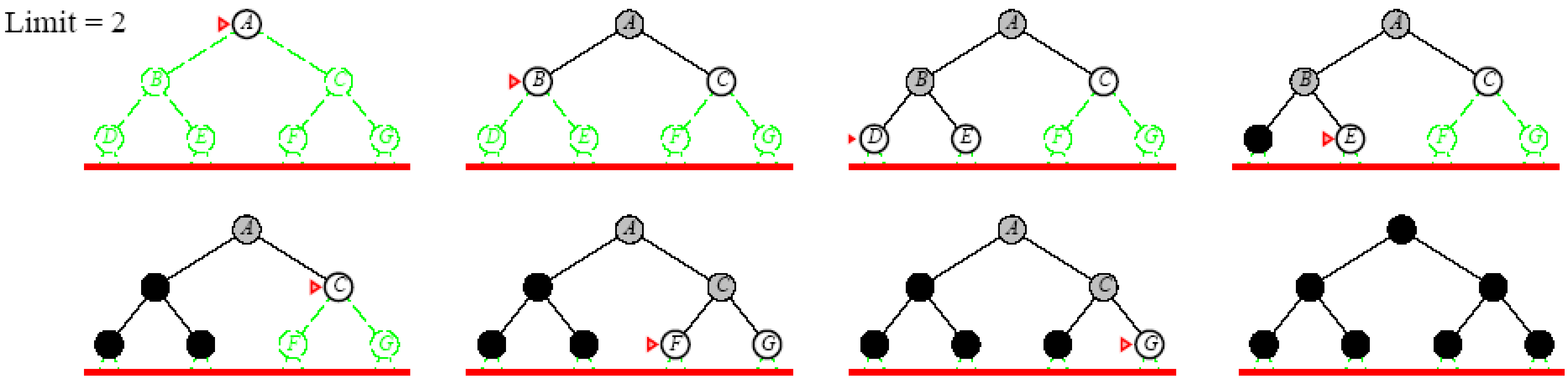
3'rd Iteration------>A, B, D, E, C, F, G

Iterative Deepening Search

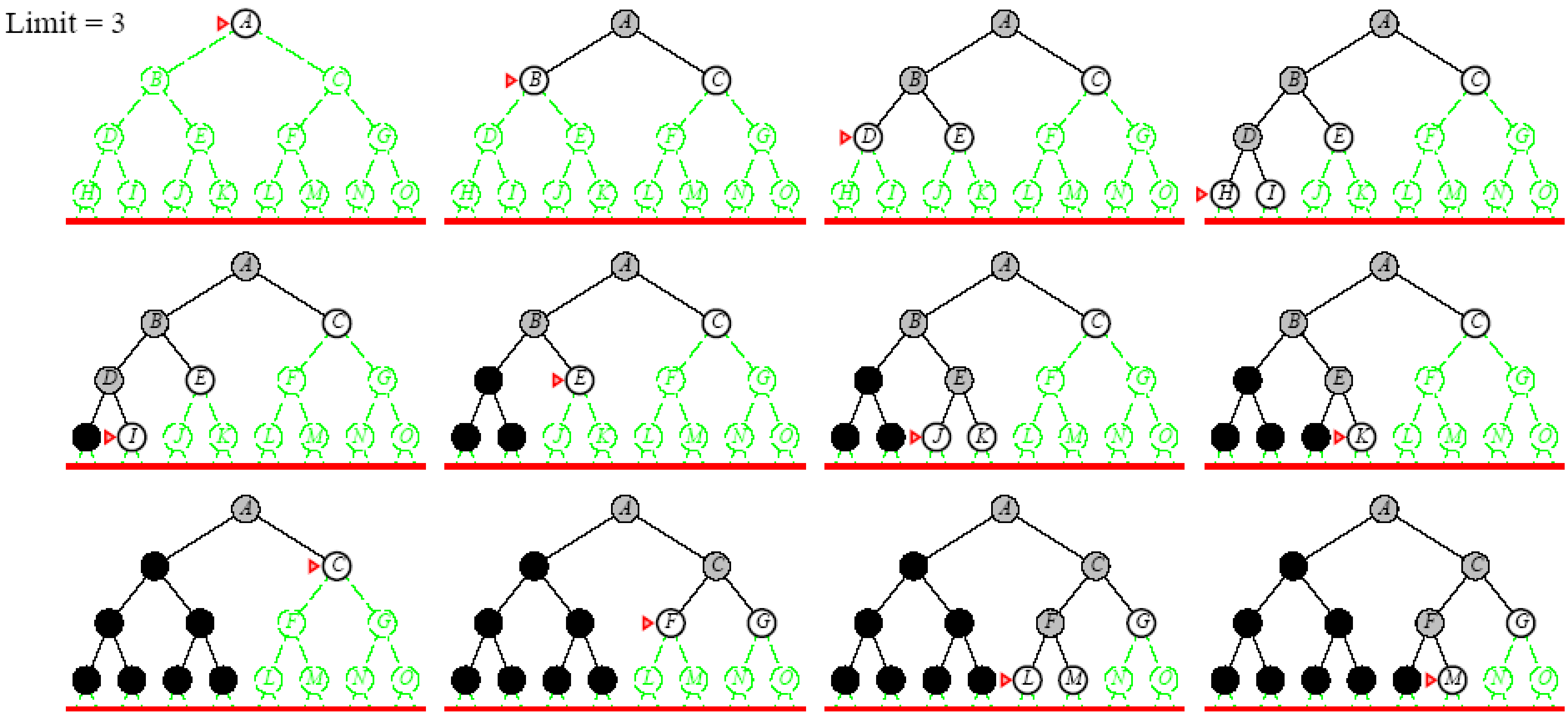
Iterative Deepening Search



Iterative Deepening Search



Iterative Deepening Search



Iterative Deepening Search

- Complete
  - Yes
- Time
  - $O(b^d)$
- Space
  - $O(bd)$
- Optimal
  - Yes if step cost = 1

Lessons From Iterative Deepening Search

- Faster than BFS even though IDS generates repeated states
  - BFS generates nodes up to level d+1
  - IDS only generates nodes up to level d

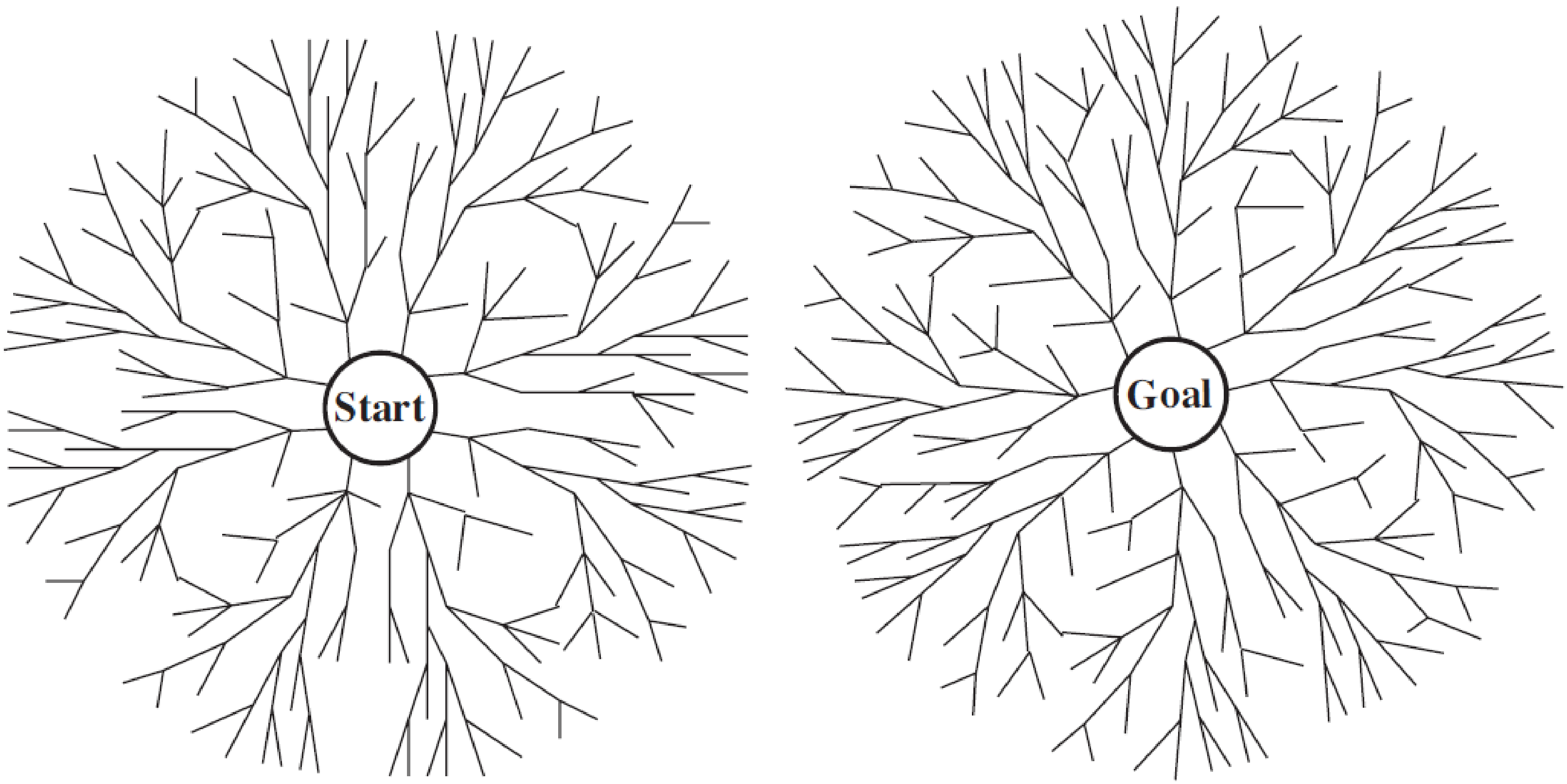
- In general, iterative deepening search is the preferred uninformed search method when there is a large search space and the depth of the solution is not known

Bidirectional Search

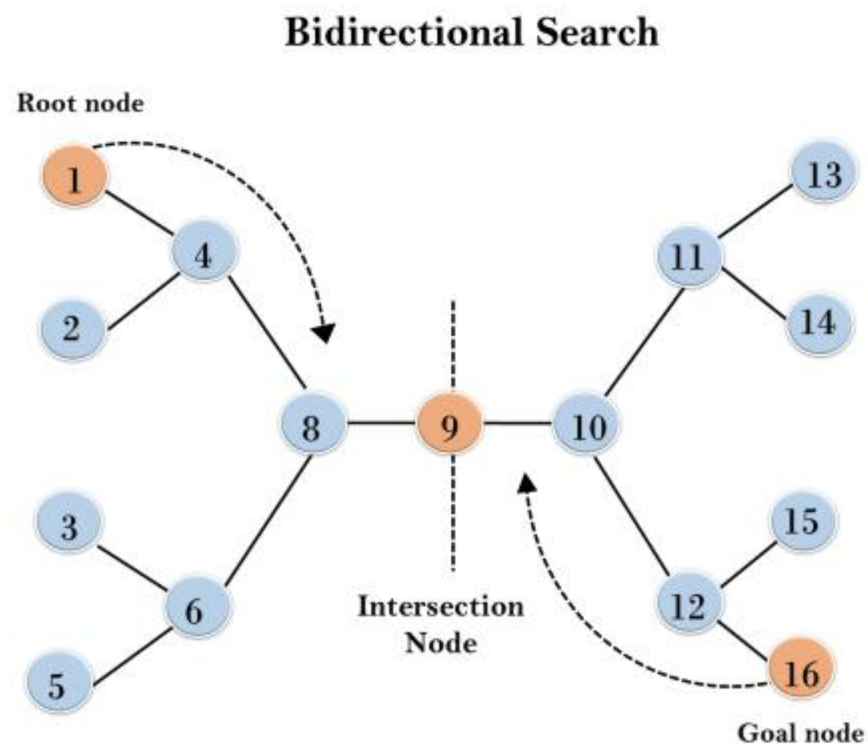
- Run two simultaneous searches
  - One forward from initial state
    - Forward Search
  - Other backward from the goal
    - Backward Search

- Divides one single search graph into two small sub-graphs
- Search Stopping Criteria?
  - When these two graphs intersect each other

Bidirectional Search



Bidirectional Search



Bidirectional Search

- Complete
  - Yes
- Time Complexity
  - $O(b^{d/2})$
- Space Complexity
  - $O(b^{d/2})$
- Optimal
  - Yes if both directions use BFS and if all step costs are identical

Bidirectional Search

- Advantages
  - Bidirectional Search is fast
  - Bidirectional search requires less memory
- Disadvantages
  - Implementation of Bidirectional search is difficult
  - Goal state should be known in advance

Comparison

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]