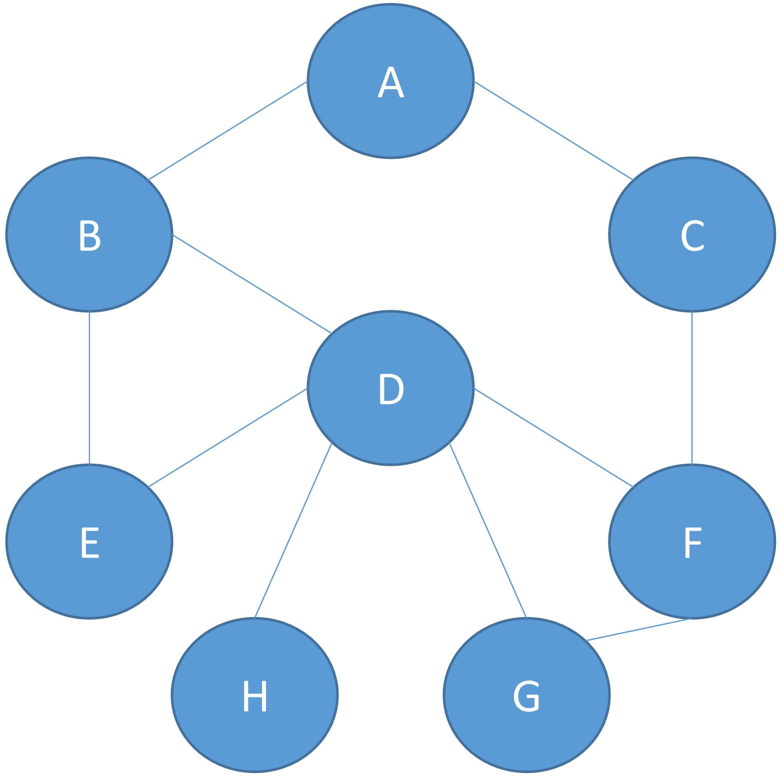
Search Techniques

Outline

- **Searching**
- **Uninformed Search Techniques**
  - **Breadth First Search**
  - **Uniform Cost Search**
  - **Depth First Search**
  - **Backtracking Search**
  - **Depth Limited Search**
  - **Iterative Deepening Depth First Search**
  - **Bidirectional Search**
  - **Search Strategy Comparison**
- Informed Search Techniques
  - Hill Climbing
  - Best First Searching
  - Greedy Search
  - A* Search
  - Adversarial Search Techniques
    - Mini-max Procedure
    - Alpha Beta Procedure

Searching

- Step in Problem Solving
- Searching is Performed through the State Space
- Searching accomplished by constructing a search tree



Searching: Steps

- Check whether the current state is the goal state or not
- Expand the current state to generate the new sets of states
- Choose one of the new states generated for search which entire depend on the selected search strategy
- Repeat the above steps until the goal state is reached or there are no more states to be expanded

Searching: Criteria to Measure Performance

- Completeness: Ability to find the solution if the solution exists
- Optimality: Ability to find out the highest quality solution among the several solutions
  - Should maintain the information about the number of steps or the path cost from the current state to the goal state
- Time Complexity: Time taken to find out the solution
- Space Complexity: Amount of Memory required to perform the searching

Searching: Types

- Blind Search or Uninformed Search
- Informed Search or Heuristic Search

Searching: Evolution Function

- A number to indicate how far we are from the goal
- Every move should reduce this number or if not never increase
- When this number becomes zero, the problem is solved (there may be some exceptions)

8 Puzzle Games

1	2	3
8		4
7	6	5

- Its Goal State
- Evolution Function = 0

2	8	3
1	4	
7	6	5

- Its Initial State
- Evolution Function = -4

Searching: Problem Classification

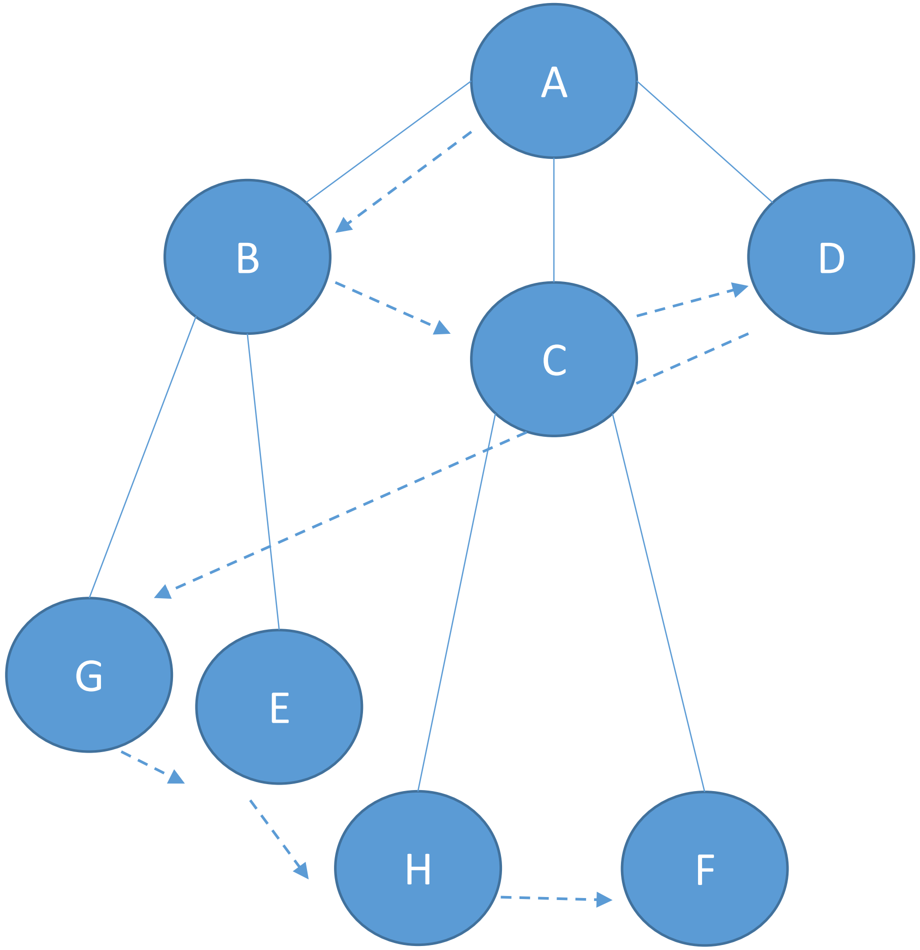
- Ignorable: Intermediate actions can be ignored. Example: Water Jug Problem
- Recoverable: The actions can be implemented to go the initial states. Example: 8 Puzzle Games
- Irrecoverable: The actions can’t be implemented to reach the previous state. Example: Tic-Tac-Toe
- Decomposable: The problem can be broken into similar ones. Example: Bike Racing

Uniformed Search

- Search provided with problem definition only and no additional information about the state space
- Expansion of current state to new set of states is possible
- It can only distinguish between goal state and non-goal state
- Less effective compared to Informed search

Breadth First Search

- Root node is expanded first
- Then all the successors of the root node are expanded
- Then their successors are expanded and so on.
- Nodes, which are visited first will be expanded first (FIFO)
- All the nodes of depth ‘d’ are expanded before expanding any node of depth ‘d+1’



Breadth First Search: Four Criteria

- Completeness
  - d: depth of the shallowest goal
  - b: branch factor
  - This search strategy finds the shallowest goal first
  - Complete, if the shallowest goal is at some finite depth
- Optimality
  - If the shallowest goal nodes were available, it would already have been reached
  - Optimal, if the path cost is a non-decreasing function of the path of the node

Breadth First Search: Four Criteria

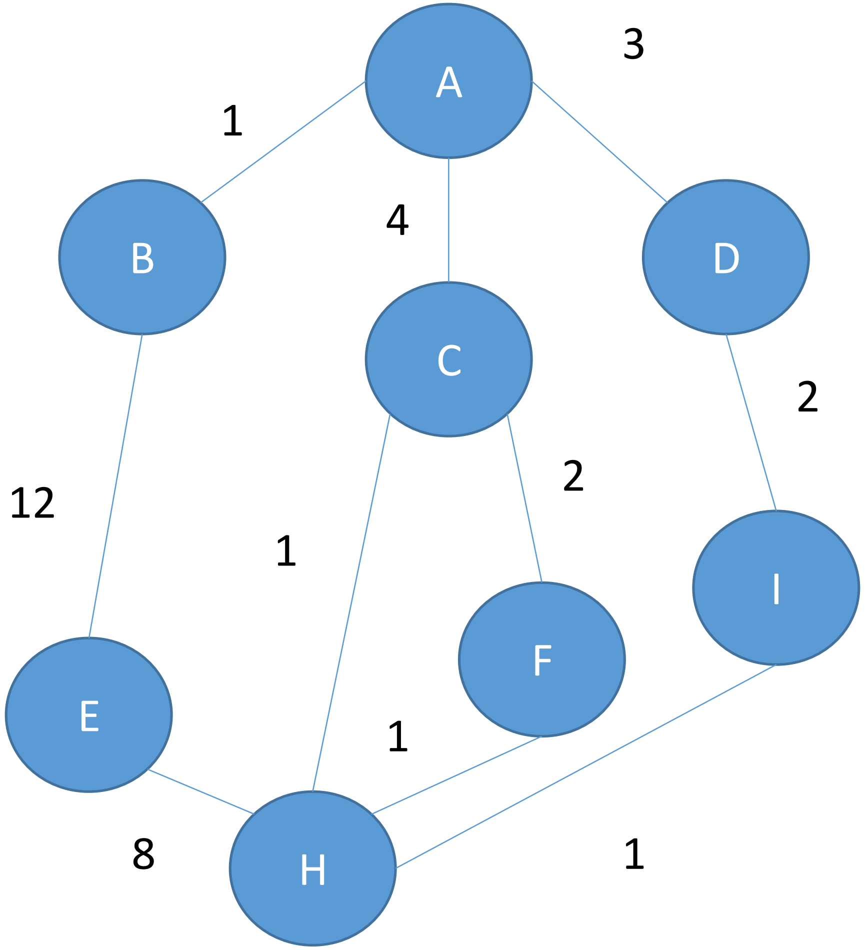
- Time Complexity
  - For a search tree a branching factor ‘b’ expanding the root yields ‘b’ nodes at the first level.
  - Expanding ‘b’ nodes at first level yields b^2 nodes at the second level.
  - Similarly, expanding the nodes at $(d+1)^{th}$ level yields b^d node at d^{th} level
  - If the goal is in d^{th} level, in the worst case, the goal node would be the last node in the d^{th} level
- Hence, We should expand (b^d-1) nodes in the d^{th} level (Except the goal node itself which doesn’t need to be expanded)
- So, Total number of nodes generated at d^{th} level $= b(b^d-1) = b^{d+1}-b$
- Again, Total number of nodes generated $= 1+b+b^2+\ldots+b^{d+1}-b = O(b^{d+1})=O(b^d)$
- Hence, time complexity is $O(b^{d+1})$ where, b= branching factor and d= level of goal node in the search table

Breadth First Search: Four Criteria

- Space Complexity
  - Same as time complexity
  - i.e. $O(b^{d+1})$
  - Since each node has to be kept in the memory
- Disadvantages
  - Memory Wastage
  - Irrelevant Operations
  - Time Intensive
  - It doesn’t assure the optimal cost solution

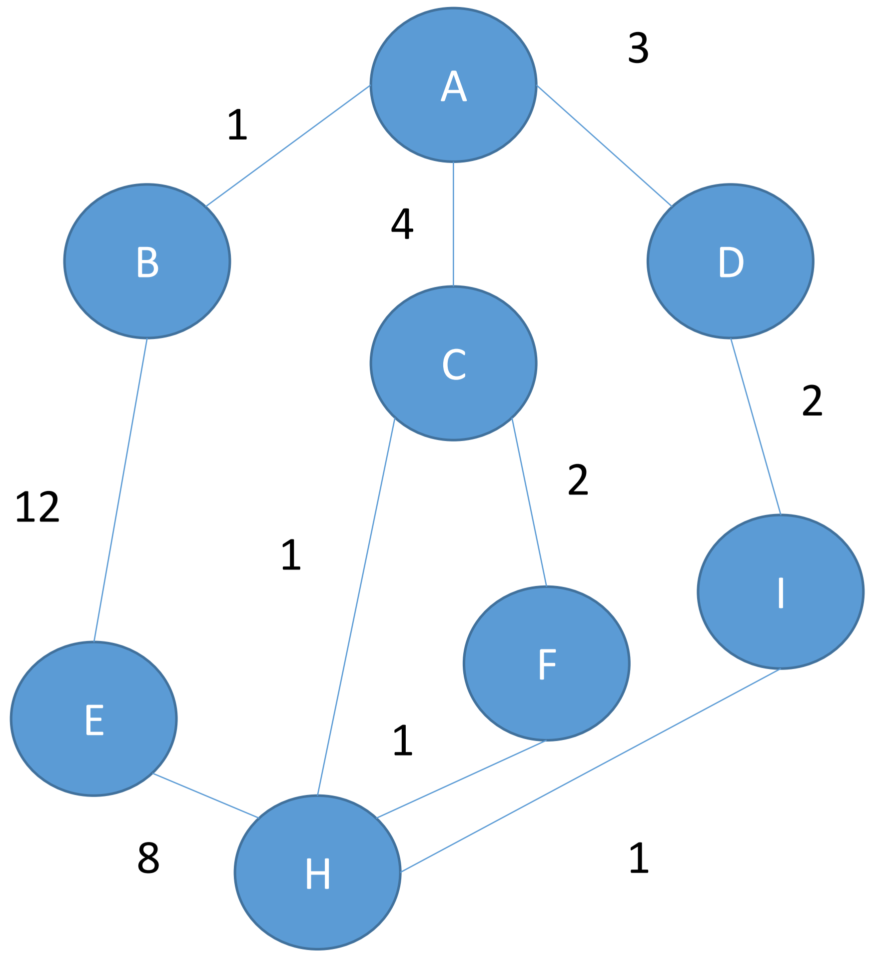
Uniform Cost Search

- It expands the lowest cost mode on the fringe
- The first solution is guaranteed to be the cheapest one because a cheaper one would have expanded earlier and so would have been found first
- Required Condition: A to H
  - ABEH=21, ACH=5, ACFH=7, ADIH=6



Uniform Cost Search

- Solution: Required Operation
  - Expand A→ Yield B, C, D
    With AB=1, AC=4, AD=3
  - Expand B→ Yield E with ABE=13
    As ABE>AC and ABE>AD
  - Expand D→ Yield I with ADI=5
    As ADI>AC
  - Expand C→ Yield H and F with
    ACH=5 and ACF=6
  - Solution Achieved
- If all step costs are equal, it is identical breadth first search



Uniform Cost Search

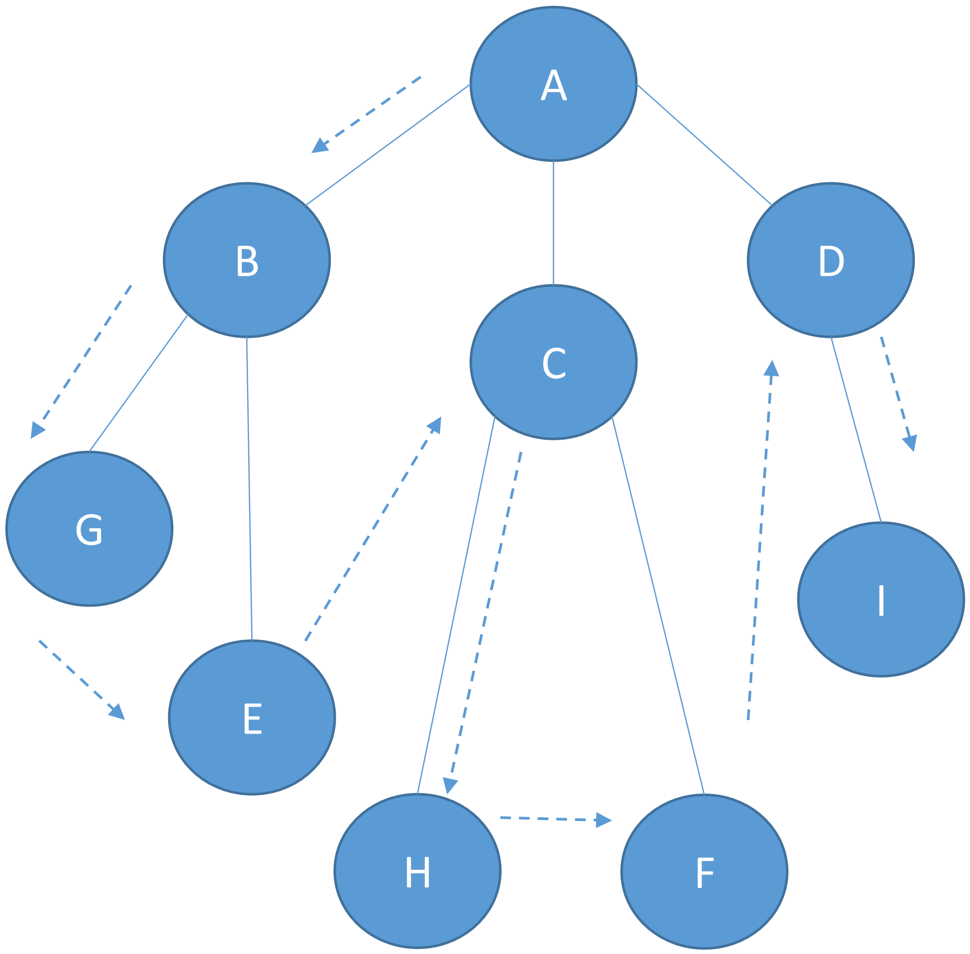
- Disadvantages
  - Doesn’t care about the number of steps a path has but only about their cost
  - It might get stuck in an infinite loop if it expands a node that has a zero cost action leading back to same state

Uniform Cost Search: Four Criteria

- Completeness
  - Complete, if the cost of every step is greater than or equal to some small positive constant E
- Optimality
  - The same ensures optimality
- Time Complexity
  - O(b $^{C^*/E}$)
  - Where C*→ cost of optimal path and E→ small positive constant
  - This complexity is much greater than that of Breadth first search
- Space Complexity
  - O(b $^{C^*/E}$)

Depth First Search

- Expands the nodes at the deepest level of the search tree (LIFO)
- When a dead end is reached, the search backup to the next node that still has unexplored successors



Depth First Search: Four Criteria

- Completeness
  - Can get stuck going down the wrong path
  - It will always continue downwards without backing up
  - If the path chose get infinitely down, even when shallow solution exists
  - Not complete
- Optimality
  - The strategy might return a solution path that is longer than the optimal solution, if it starts with an unlucky path
  - Not optimal

Depth First Search: Four Criteria

- Space Complexity
  - It needs to store a single path from root to a leaf node and the remaining unexpanded sibling nodes for each node in the path
  - For a search tree of branching factor ‘b’ and maximum tree depth ‘m’, only the storage of b_{m+1} node is required
  - Hence,
    Space Complexity= O(b.m+1)
    = O(bm)
- Time Complexity
  - $O(b^m)$, in the worst case, since in the worst case all the b^m nodes of the search tree would be generated
  - Hence,
    Time Complexity= $O(b^m)$

Backtracking Search

- It uses still less memory
- Only one successor is generated at a time rather than all
- Each partially expanded nodes remember which node to expand next

- Completeness: Not Complete
- Optimality: Not Optimal
- Time Complexity= $O(b^m)$
- Space Complexity= $O(m)$

Depth Limited Search

- Modification of depth first search
- Depth first search with predetermined limit ‘l’
- After the nodes at the level ‘l’ are explored, the search backtracks without going further deep
- Hence, it solves the infinite path problem of the depth first search strategy
- Completeness: Complete except at additional source of incompleteness if l>d
- Optimality: Optimal except at l>d
- Time Complexity=$O(b^l)$
- Space Complexity=O(bl)

Iterative Deepening Depth First Search

- Finds the best limit by gradually increasing depth limit l first to 0, then to 1, 2 and so on
- Combines the benefits of the depth first and breadth first search
- The complex part is to choose good depth limit
- This strategy addresses the issue of good depth limit by trying all possible depth limits
- The process is repeated until goal is found at depth limit ‘d’ which is the depth of shallowest goal
- Completeness: as of Breadth First Search i.e. Complete if branching factor is finite
- Optimality: as of Breadth First Search i.e. optimal if the path cost is non decreasing function of depth
- Time Complexity= $O(b^d)$
- Space Complexity= $O(b^d)$

Bidirectional Search

- Performs two simultaneous searches, one forward from initial state and the other backward from the last state
- Search stops when the two traversals meet in the middle
- Completeness: Complete if both searches are B.F.S. and b is finite
- Optimality: Optimal if both searches are B.F.S.
- Time Complexity
  - For B.F.S. is $O(b^{d+1})$
  - If B.F. Bidirectional Search is used then the complexity = $O(b^{d/2})$
  - Since the forward and backward searches have to go halfway only
- Space Complexity= $O(b^{d/2})$

Informed Search

References

- Russell, S. and Norvig, P., 2011, Artificial Intelligence: A Modern Approach, Pearson, India.
- Rich, E. and Knight, K., 2004, Artificial Intelligence, Tata McGraw hill, India.

Thank You

Any Queries?

Use “Search technique” so that you could find the optimal solution within yourself.