Elective Course
Course Code: CS4103
Autumn 2025-26

Lecture #07

Artificial Intelligence for Data Science

Week-2: PROBLEM SOLVING BY SEARCH

Introduction to Uninformed Search Techniques [Part-II]

Course Instructor:

Dr. Monidipa Das
Assistant Professor
Department of Computational and Data Sciences
Indian Institute of Science Education and Research Kolkata, India 741246

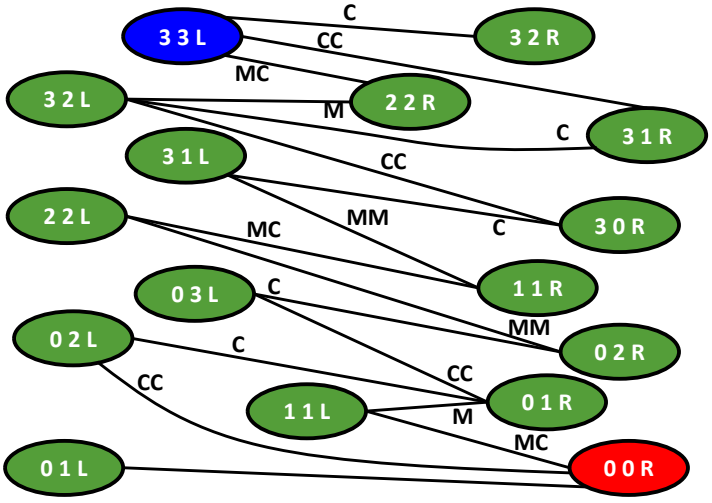
Graph Search

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

✓GRAPH-SEARCH algorithm contains at most one copy of each state

✓the frontier separates the state-space graph into the explored region and the unexplored region, so that every path from the initial state to an unexplored state has to pass through a state in the frontier.

Missionaries and Cannibals Problem Solution: DFS [GRAPH-SEARCH]



L={(33L)}
E={}

L={(32R),(31R),(22R)}
E={(33L)}

L={(31R),(22R)}
E={(33L),(32R)}

L={(32L),(22R)}
E={(33L),(32R),(31R)}

L={(30R),(22R)}
E={(33L),(32R),(31R),(32L)}

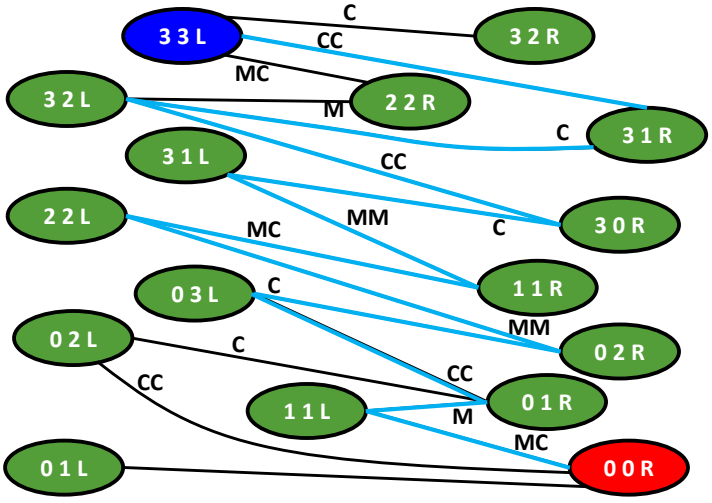
L={(31L),(22R)}
E={(33L),(32R),(31R),(32L),(30R)}

L={(11R),(22R)}
E={(33L),(32R),(31R),(32L),(30R),(31L)}

L={(22L),(22R)}
E={(33L),(32R),(31R),(32L),(30R),(31L),(11R)}

L={(02R),(22R)}
E={(33L),(32R),(31R),(32L),(30R),(31L),(11R),(22L)}

Missionaries and Cannibals Problem Solution: DFS [GRAPH-SEARCH]



L={(03L),(22R)}
E={(33L),(32R),(31R),(32L),(30R),(31L),(11R),(22L),(02R)}

L={(01R),(22R)}
E={(33L),(32R),(31R),(32L),(30R),(31L),(11R),(22L),(02R),(03L)}

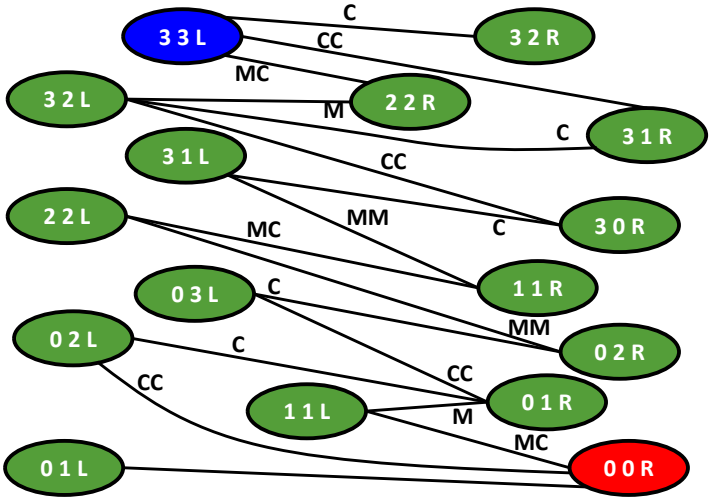
L={(11L),(02L),(22R)}
E={(33L),(32R),(31R),(32L),(30R),(31L),(11R),(22L),(02R),(03L),(01R)}

L={(00R),(02L),(22R)}
E={(33L),(32R),(31R),(32L),(30R),(31L),(11R),(22L),(02R),(03L),(01R),(11L)}

G (goal state) found!

Soln: CC C CC C MM MC MM C CC M MC

Missionaries and Cannibals Problem Solution: BFS [GRAPH-SEARCH]



L={(33L)}
E={}

L={(22R),(31R),(32R)}
E={(33L)}

L={(31R),(32R), (32L)}
E={(33L),(22R)}

L={(32R), (32L)}
E={(33L),(22R),(31R)}

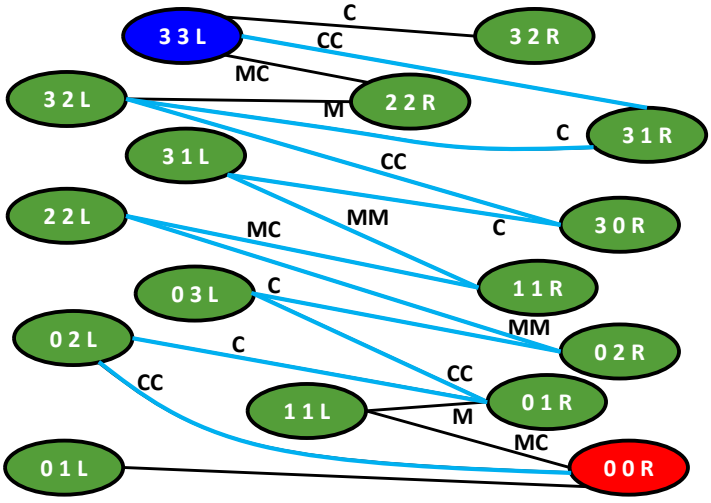
L={(32L)}
E={(33L),(22R),(31R),(32R)}

L={(30R)}
E={(33L),(22R),(31R),(32R),(32L)}

L={(31L)}
E={(33L),(22R),(31R),(32R),(32L),(30R)}

L={(11R)}
E={(33L),(22R),(31R),(32R),(32L),(30R),(31L)}

Missionaries and Cannibals Problem Solution: BFS [GRAPH-SEARCH]



Soln: CC C CC C MM MC MM C CC C CC

L={(22L)}
E={(33L),(22R),(31R),(32R),(32L),(30R),(31L),(11R)}

L={(02R)}
E={(33L),(22R),(31R),(32R),(32L),(30R),(31L),(11R),(22L)}

L={(03L)}
E={(33L),(22R),(31R),(32R),(32L),(30R),(31L),(11R),(22L),(02R)}

L={(01R)}
E={(33L),(22R),(31R),(32R),(32L),(30R),(31L),(11R),(22L),(02R),(03L)}

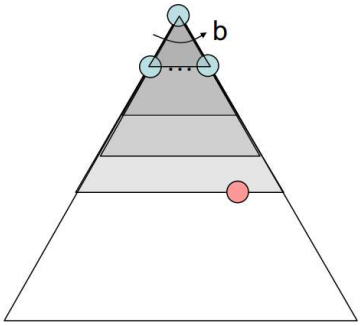
L={(02L),(11L)}
E={(33L),(22R),(31R),(32R),(32L),(30R),(31L),(11R),(22L),(02R),(03L),(01R)}

L={(11L),(00R)}
E={(33L),(22R),(31R),(32R),(32L),(30R),(31L),(11R),(22L),(02R),(03L),(01R),(02L)}

L={(00R)}
E={(33L),(22R),(31R),(32R),(32L),(30R),(31L),(11R),(22L),(02R),(03L),(01R),(02L),(11L)} **G (goal state) found!**

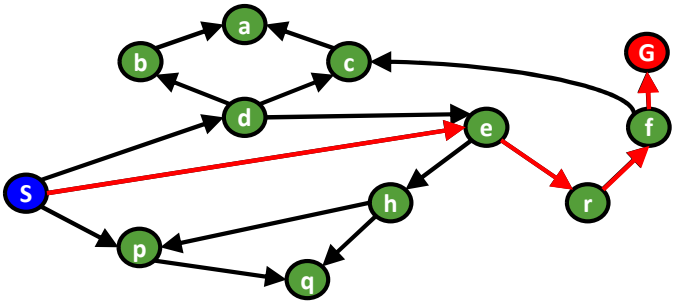
Iterative Deepening Depth-First Search

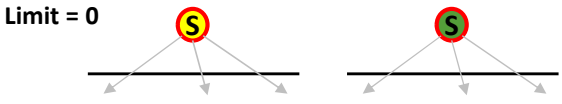
- **Idea:** get DFS's space advantage with BFS's time / shallow-solution advantages
  - Run a DFS with depth limit 0. If no solution…
  - Run a DFS with depth limit 1. If no solution…
  - Run a DFS with depth limit 2. If no solution…
  - Run a DFS with depth limit 3. …..

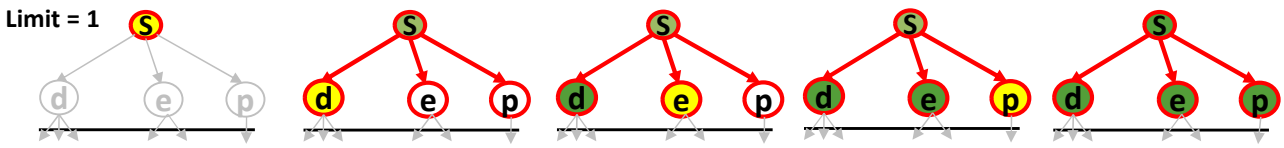


- **Isn't that wastefully redundant?**
  - Generally most work happens in the deepest level searched, so not so bad

Iterative Deepening Depth-First Search

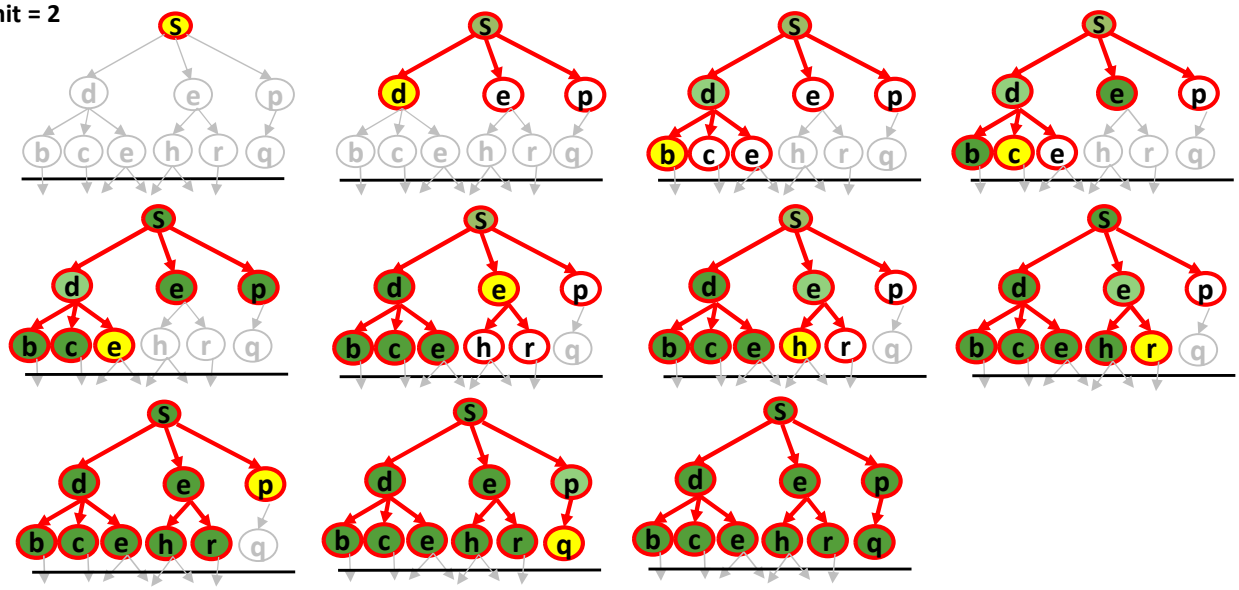






Iterative Deepening Depth-First Search

Limit = 2



Iterative Deepening Depth-First Search

Complete? Yes

Time complexity? $db^1 + (d-1)b^2 + \ldots + b^d = O(b^d)$

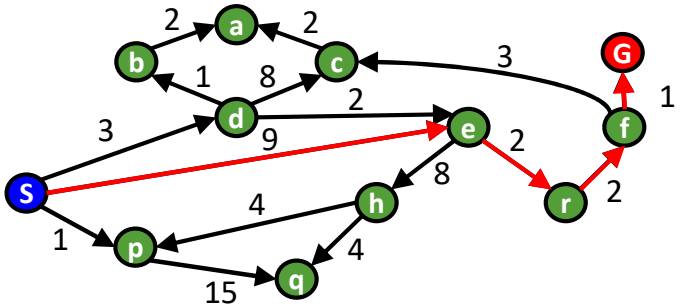
Space complexity? $O(bd)$

Optimal? Only if step costs are all identical

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

Optimality of Breadth-First Search

- **Is BFS optimal?**
  - It depends
  - Breadth-first search always finds the shallowest goal state
  - The path to that goal state, however, may have a higher cost than one to a deeper goal state

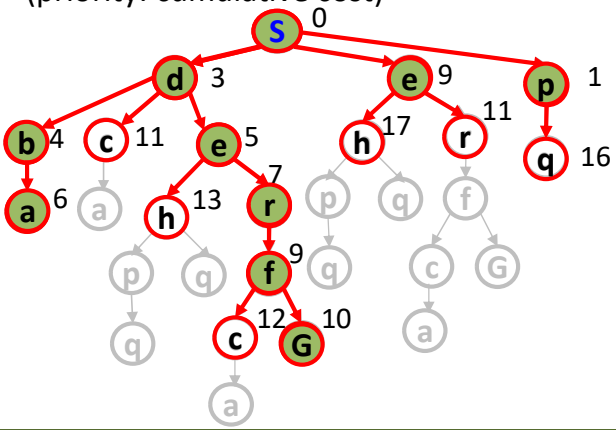


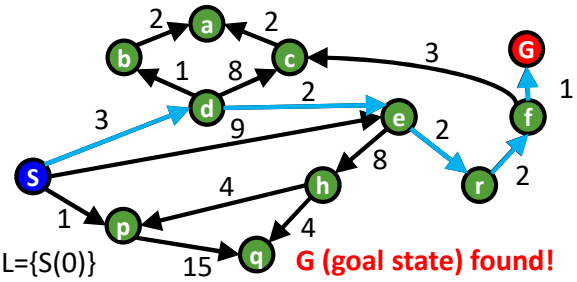
- **If we are looking for least-cost solutions, breadth-first is suboptimal** unless all step costs are identical

✓ BFS finds the shortest path in terms of number of actions.
✓ It does not find the least-cost path.

Uniform-Cost Search (UCS)

- **Strategy:** Expand least-cost unexpanded node or the cheapest node
- **Implementation:** Fringe = **priority queue** (priority: cumulative cost)





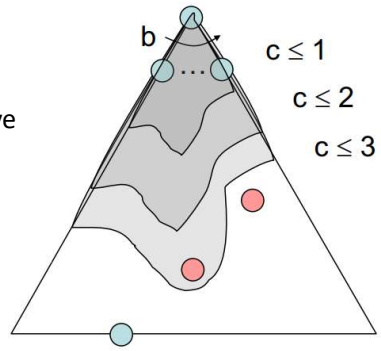
L={S(0)}
L={p(1), d(3), e(9)}
L={d(3), e(9), q(16)}
L={b(4), e(5), e(9), c(11), q(16)}
L={e(5), a(6), e(9), c(11), q(16)}
L={a(6), r(7), e(9), c(11), h(13), q(16)}
L={r(7), e(9), c(11), h(13), q(16)}
L={e(9), f(9), c(11), h(13), q(16)}
L={f(9), c(11), r(11), h(13), q(16), h(17)}
L={G(10), c(11), r(11), c(12), h(13), q(16), h(17)}

G (goal state) found!

Path:
S→ d→ e→ r→ f→G

Uniform-Cost Search (UCS)

- **What nodes does UCS expand?**
  - Processes all nodes with cost less than cheapest solution!
  - If that solution costs C* and arcs cost at least ϵ, then the "effective depth" is roughly C*/ϵ
  - Takes time O($b^{C^*/\epsilon}$) (exponential in effective depth)
- **How much space does the fringe take?**
  - O($b^{C^*/\epsilon}$)
- **Is it complete?**
  - Assuming best solution has a finite cost and minimum arc cost is positive, yes!
- **Is it optimal?**
  - Yes!



When all step costs are equal, UCS is similar to BFS. However, BFS can stop as soon as it generates a goal, whereas UCS examines all the nodes at the goal's depth to see if one has a lower cost.

Comparing Uninformed Search Strategies

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

b, branching factor — d, depth of shallowest solution

m, depth of search tree — C^*, cost of optimal solution

[a] if b is finite [b] is step costs $> \epsilon$ for some $\epsilon > 0$ [c] is all step costs are the same

When to use what

- **Depth-First Search:**
  - Solutions are far from the initial state
- **Breadth-First Search**:
  - Some solutions are known to be shallow (not deep).
- **Uniform-Cost Search**:
  - Actions have varying costs
  - Least cost solution is required

  This is the only uninformed search that worries about costs.
- **Iterative-Deepening Search**:
  - Computational space is limited and the shortest solution path is required

Questions?