Final Assessment│Spring 2023

Md. Shafayet Hossain

CSE - 21st Batch │ **Course Title:** Artificial Intelligence

Course Code: CSI - 341 │ **ID:** 2121210071

Answer to the Question no- 1

(a)

Blind Search Algorithm-

A blind search (also called an uninformed search) is a search that has no information about its domain. The only thing that a blind search can do is distinguish a non-goal state from a goal state.

List of Blind Search Algorithm-

- Breadth – first Search
- Uniform – cost Search
- Depth - first Search
- Depth - limited Search
- Iterative deepening Depth-first Search

Blind Search Algorithm Properties-

Breadth – first Search:

- Complete? Yes (if *b* is finite)
- Time? $1+b+b^2+b^3+\ldots+b^d + b(b^d-1) = O(b^{d+1})$
- Space? $O(b^{d+1})$ (keeps every node in memory)
- Optimal? Yes (if cost = 1 per step)
- Space is the bigger problem (more than time)

Uniform – cost Search:

- Complete? Yes, if step cost ≥ ε (cost per action)
- Time? # of nodes with *g* ≤ cost of optimal solution, ε $O(b^{ceiling(C^*/\varepsilon)})$ where C^* is the cost of the optimal solution
- Space? # of nodes with *g* ≤ cost of optimal solution, $O(b^{ceiling(C^*/\varepsilon)})$
- Optimal? Yes – nodes expanded in increasing order of *g(n)*

Depth - first Search:

- Complete? No: fails in infinite-depth spaces, spaces with loops
    - Modify to avoid repeated states along path
        - → complete in finite spaces
- Time? $O(b^m)$: terrible if *m* is much larger than *d*
    - but if solutions are dense, may be much faster than breadth-first
- Space? *O(bm)*, i.e., linear space!
- Optimal? No

Depth - limited Search:

= depth-first search with depth limit *l*,

i.e., nodes at depth *l* have no successors

Iterative deepening Depth-first Search:

- Complete? Yes
- Time? $(d+1)b^0 + d\,b^1 + (d-1)b^2 + \ldots + b^d = O(b^d)$
- Space? *O(bd)*
- Optimal? Yes, if step cost = 1

(b)

- **Four general steps in problem solving:**
  - Goal formulation
    - What are the successful world states
  - Problem formulation
    - What actions and states to consider given the goal
  - Search
    - Determine the possible sequence of actions that lead to the states of known values and then choosing the best sequence.
  - Execute
    - Give the solution perform the actions.

Answer to the Question no- 2

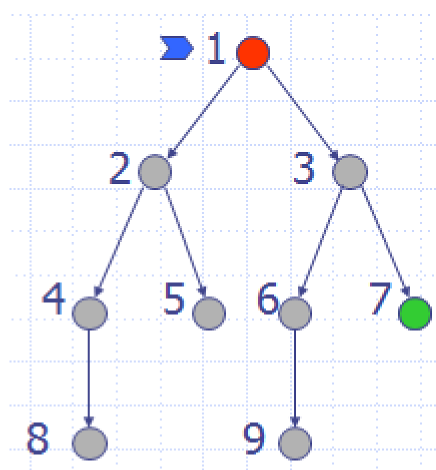
(a)

States of a Robot Assembly-

- **States:** Real-valued coordinates of robot joint angles; parts of the object to be assembled.
- **Initial state:** Any arm position and object configuration.
- **Actions:** Continuous motion of robot joints
- **Goal test:** Complete assembly (without robot)
- **Path cost:** Time to execute

(b)

States for the given tree using BFS search-strategy:



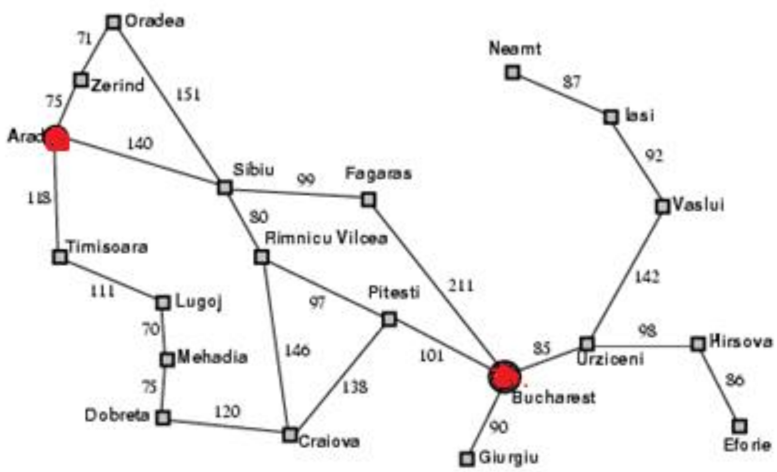
FRINGE = (1)

- Expand *shallowest* unexpanded node
- Implementation: *fringe* is a FIFO queue
- New nodes are inserted at the end of the queue

Answer to the Question no- 4

(a)

Problem Solving Agent-



- On holiday in Romania; currently in Arad.
- Flight leaves tomorrow from Bucharest
- Formulate goal:
    - be in Bucharest
- Formulate problem:
    - states: various cities
    - actions: drive between cities
- Find solution:
    - sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest

(b)

Limitations of DFS Strategy-

- Depth-first with depth cutoff k (maximal depth below which nodes are not expanded)
- Three possible outcomes:
  - Solution
  - Failure (no solution)
  - Cut-off (no solution within cutoff)
  - Solves the infinite-path problem.
- If $k < d$ then incompleteness results.
- If $k > d$ then not optimal.
- Time complexity: $O(b^k)$
- Space complexity O(bk)

Answer to the Question no- 5

(a)

Avoiding Repeated States-

- Depth-first strategy:
  - Solution 1:
    - Keep track of all states associated with nodes in current tree
    - If the state of a new node already exists, then discard the node

→ Avoids loops

  - Solution 2:
    - Keep track of all states generated so far
    - If the state of a new node has already been generated, then discard the node

→ Space complexity of breadth-first

(b)

Real-world Problems where we can use searching algorithms-

Seartching algorithm is used in wide sectors around the world. Some real world problems where searching algorithm is used are given below:

- Route finding
- Touring problems
- VLSI layout
- Robot Navigation
- Automatic assembly sequencing
- Drug design
- Internet searching