Artificial Intelligence (AI)
CS-3011
Evaluation Scheme for the Mid-Semester Examination (Autumn 2023)

Q1.

(a) Who is Alan Turing? Has ChatGPT-3 passed the Turing test?

Alan Turing was a British mathematician, logician, and computer scientist. proposed the Turing Test in his 1950 paper titled "Computing Machinery and Intelligence." This test is a benchmark for determining a machine's ability to exhibit human-like intelligence. [Mark 1]

As of my last knowledge update in September 2021, ChatGPT-3, which is based on the GPT-3.5 architecture, had not definitively passed the Turing Test. Passing the Turing Test is a complex and controversial benchmark for assessing a machine's ability to exhibit human-like intelligence and understanding in natural language conversations.

However, the Turing Test's criteria and standards can vary, and opinions on what constitutes "passing" can differ among researchers and experts. It's also important to note that **the Turing Test is just one way to assess the capabilities of AI systems**, and many in the field are working on developing more comprehensive and nuanced evaluation methods. (Source: ChatGPT-3.5)
[Mark 1 for No, also 1 for Yes but justification is required; step marking as per faculty discretion]

(b) Describe the types of searching techniques—uninformed search and informed search. Why is "searching" important in artificial intelligence? Justify your answer.

Searching is a fundamental concept in artificial intelligence (AI) and is essential for problem-solving and decision-making. There are two main categories of searching techniques in AI: uninformed search and informed search. These techniques are used to navigate through problem spaces and find solutions to various AI problems.

Uninformed Search (Blind Search):

Breadth-First Search (BFS): This technique explores all the neighbor nodes at the current depth level before moving on to the next level. It guarantees finding the shallowest solution, but it may not be efficient in terms of memory usage.

Depth-First Search (DFS): DFS explores as far as possible along a branch before backtracking. It can be memory-efficient but may not guarantee finding the optimal solution in terms of depth.

Uniform-Cost Search (UCS): UCS considers the cost of reaching each node and explores the lowest-cost path first. It is used in problems with different path costs.

Depth-Limited Search (DLS): DLS is a variation of DFS that limits the depth of the search to avoid infinite loops in large state spaces.

Informed Search (Heuristic Search):

A* Search: A* combines the cost to reach a node (g) with a heuristic estimate of the cost to the goal (h). It explores nodes with the lowest f = g + h value first. A* is complete and optimal if the heuristic is admissible (never overestimates the true cost).

Greedy Best-First Search: This search algorithm selects the node that is estimated to be closest to the goal according to the heuristic. It does not guarantee optimality.

Iterative Deepening A (IDA):** IDA* combines depth-first search's memory efficiency with A*'s optimality by iteratively increasing the depth limit.

Why Is "Searching" Important in Artificial Intelligence?

Searching is a fundamental concept in AI, and it is crucial for several reasons:

Problem Solving: AI systems often face complex problems that require searching through large state spaces to find solutions. Searching techniques provide a systematic way to explore and discover these solutions.

Decision Making: Many AI applications involve making decisions based on available information and constraints. Searching helps in evaluating various options and selecting the best course of action.

Optimization: Searching is essential for optimization problems where the goal is to find the best possible solution among a set of alternatives. Informed search techniques like A* can efficiently find optimal solutions.

Artificial Intelligence (AI)
CS-3011
Evaluation Scheme for the Mid-Semester Examination (Autumn 2023)

Route Planning: In applications like GPS navigation, robotics, and logistics, searching algorithms are used to find the most efficient routes or paths from one location to another.

Game Playing: AI agents in games like chess, Go, and video games use search algorithms to explore possible moves and make strategic decisions.

(c) Natural Language Processing: In language understanding and generation tasks, searching is employed to find relevant information in large corpora of text or to generate coherent and contextually relevant responses.

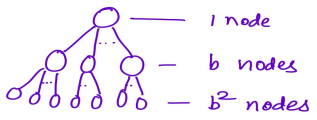
(d)

(e) In summary, searching techniques are essential in AI because they enable machines to navigate complex problem spaces, make informed decisions, optimize solutions, and solve a wide range of real-world problems efficiently and effectively. The choice of search algorithm depends on the specific problem and its requirements, whether it's finding a path, making decisions, or optimizing solutions.

[Mark 1 + 1, step marking as per faculty discretion]

(c) Doubling your computer's speed allows you to double the depth of a tree search given the same amount of time—is this true or false? Give a brief explanation of your answer.

FALSE.



Suppose initially computer can process search tree till depth d in time T. so no. of nodes can be processed in T time are

$$N_d = 1 + b + b^2 + \cdots + b^d$$

Now if we double computer's speed, it can process twice the number of nodes compared to before i.e. it can process $2N_d$ nodes now in time T.

But if we check the number of nodes in next depth $d+1$ then –

$$N_{d+1} = 1 + b + b^2 + \cdots + b^{d+1}$$

which is much larger, compared to $2 \cdot N_d$.

$N_{d+1} >> 2 \cdot N_d$ which means that even if we double computer's speed it won't be able to process next level itself so doubling search tree's depth is far from it since $2d$ depth will have

$$N_{2d} = 1 + b + b^2 + \cdots + b^d + \cdots b^{2d} \text{ nodes.}$$

Therefore answer will be FALSE.

[Mark 1 + 1, step marking as per faculty discretion]

(d) Explain analytically, why IDS is a better uninformed search strategy as compared to BFS for the same b (branching factor) and d (depth of shallowest goal node). Assume the values for b and d.

Artificial Intelligence (AI)
CS-3011
Evaluation Scheme for the Mid-Semester Examination (Autumn 2023)

To compare Iterative Deepening Search (IDS) and Breadth-First Search (BFS) as uninformed search strategies for the same branching factor (b) and depth of the shallowest goal node (d), let's analyze the performance of both algorithms.

Breadth-First Search (BFS):
In BFS, all nodes at a given depth level are explored before moving on to the next level.
The maximum number of nodes stored in memory at any given time can be substantial. It can be on the order of b^d, where b is the branching factor, and d is the depth of the shallowest goal node.
BFS is guaranteed to find the shallowest goal, making it complete and optimal when all step costs are equal.

Iterative Deepening Search (IDS):
IDS is a depth-limited search that performs depth-first search (DFS) iteratively with increasing depth limits.
In the worst case, IDS explores nodes to a depth limit of d, then to a depth limit of d+1, and so on until the shallowest goal node is found. The maximum number of nodes stored in memory at any given time is the number of nodes at the current depth limit. Therefore, the memory requirement is limited to b^d (at the deepest level of the search). IDS is complete and optimal as long as the branching factor is finite and the cost function is non-decreasing with depth. This means that if a solution exists, IDS will find it and will find the shallowest goal node.

Analytical Comparison:
Now, let's compare BFS and IDS in terms of memory usage and completeness:

Memory Usage:
BFS stores all nodes at a given level in memory, which can be substantial, especially if the branching factor (b) and depth (d) are large. Memory usage can be on the order of b^d.
IDS, on the other hand, only stores nodes up to the current depth limit. At the deepest level (d), it stores a maximum of b^d nodes.

Completeness:
Both BFS and IDS are complete algorithms. They will find a solution if one exists, and they will find the shallowest goal node.

In terms of memory usage, IDS is more memory-efficient than BFS for the same branching factor (b) and depth of the shallowest goal node (d). This is because IDS only keeps nodes up to the current depth limit in memory, while BFS stores all nodes at the current level. However, it's important to note that IDS achieves this memory efficiency by performing multiple depth-limited searches iteratively. This means that IDS may revisit nodes at each depth level multiple times. In contrast, BFS explores each level only once.

So, while IDS is more memory-efficient, it may have a slightly higher time complexity due to repeated exploration of nodes at different depth limits. The choice between BFS and IDS depends on the specific problem, memory constraints, and the trade-off between memory and time complexity.
[Mark 1 + 1, step marking as per faculty discretion]

(e) Justify why consistent heuristic is known as triangle heuristic? How is optimality of A* search linked to Admissible heuristic and Consistent heuristic?

Next Page

Artificial Intelligence (AI)
CS-3011
Evaluation Scheme for the Mid-Semester Examination (Autumn 2023)

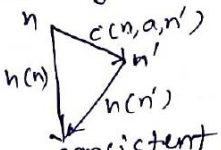
Q1(e) Justify why consistent heuristic is known as triangle inequality? How is optimality of A* search linked to Admissible heuristic and Consistent heuristic? — [2]

Ans.

- A consistent heuristic is required only for application of A* search using graph search
- A heuristic h(n) is consistent if, for every node n and every successor n' of n generated by an action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n'. Analytically this means:

$$h(n) \leq c(n, a, n') + h(n')$$

- This is the form of the general triangle inequality which stipulates that each side of a triangle can not be longer than the sum of the other two side. Here, the triangle is formed by n, n' and goal G in closest to n:



- This is why consistent heuristic is known as triangle inequality.
- Finally, the tree-search version of A* is optimal if h(n) is admissible, while the graph search version is optimal if h(n) is consistent.

[Mark 1 + 1, step marking as per faculty discretion]

Artificial Intelligence (AI)
CS-3011
Evaluation Scheme for the Mid-Semester Examination (Autumn 2023)

Q2. (a) Briefly explain the performance measure criteria for search algorithms. Discuss those criteria for the following searching algorithms
I.Depth-first Search
II.Depth-limited Search
III.Depth-first Iterative Deepening Search

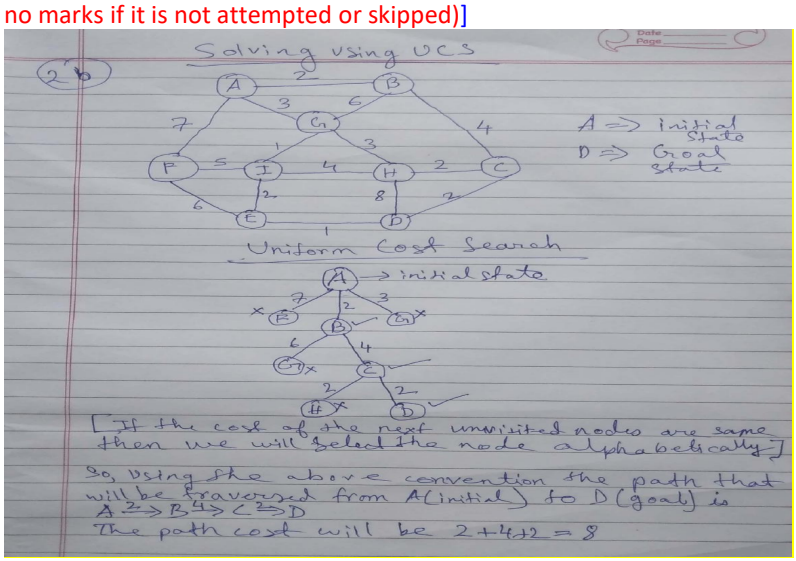
We can evaluate an algorithm's performance in four ways:
1. Completeness: It determines if the algorithm is guaranteed to find a solution when there is one.
2.Optimality: It states if the search algorithm would find the optimal solution.
3.Time complexity: It specifies the time it takes to find a solution.
4.Space complexity: This specifies the amount of memory needed to perform the search.

The typical measure of the time and space complexity is the size of the state space graph, |V | + |E|, where V is the set of vertices (nodes) of the graph and E is the set of edges (links).
I)Depth-first Search :
Completeness: No
Optimality: No
Time complexity: $O(b^m)$, where m is the maximum depth of any node and b is the branching factor.
Space complexity: O(bm)

II)Depth-limited Search :
Completeness: No
Optimality: No
Time complexity: $O(b^l)$, where l is the depth limit and b is the branching factor.
Space complexity: O(bl)

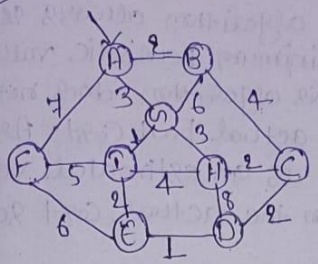
III)Depth-first Iterative Deepening Search :
Completeness: Yes if b is finite
Optimality: Yes if step costs are all identical
Time complexity: $O(b^d)$, where d is the depth and b is the branching factor.
Space complexity: O(bd)
[Marks 2 + 3, step marking as per faculty discretion]

Q2. (b) Consider the following undirected weighted graph. Apply greedy Best-First Search algorithm to find the shortest path and cost between the initial state 'A' and goal state 'D'. Also, briefly explain why the Best-First search is considered "greedy".
[Marks 5 if it solved either using UCS or Best-first Search with assumed heuristic values or justified logically it cannot be solved without heuristic values, step marking as per faculty discretion (However, no marks if it is not attempted or skipped)]



Artificial Intelligence (AI)
CS-3011
Evaluation Scheme for the Mid-Semester Examination (Autumn 2023)

Q2 b)

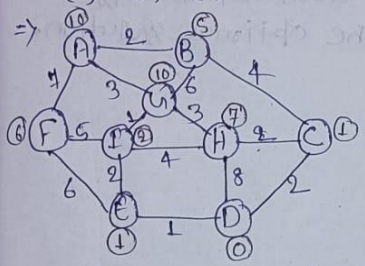


Method-1 (By ~~using~~ assuming heuristic values)

let, the heuristic values from all nodes are as follows:

$h(A) = 10$, $h(B) = 5$, $h(C) = 1$, $h(D) = 0$, $h(E) = 1$, ~~$h(F)$~~

$h(F) = 6$, $h(G) = 10$, $h(H) = 7$, $h(I) = 2$

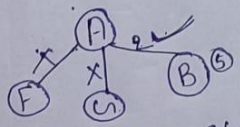


⇒ Step-I (Root node is selected)

⇒ Step-II: $f(n) = h(n)$

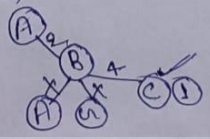
$f(B) = 5$, $f(G) = 10$, $f(F) = 6$

Node B is selected as it has minimum $f(n)$



⇒ Step-III: $f(A) = 10$, $f(G) = 10$, $f(C) = 1$

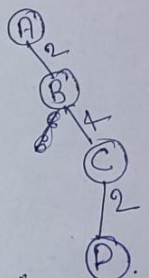
Node C is selected.



Step-IV:

$f(H) = 7$, $f(B) = 5$, $f(D) = 0$

Node D is selected



⇒ D is the goal node. Therefore, path according to greedy best first search is,

$A \rightarrow B \rightarrow C \rightarrow D$

⇒ cost is,

$2 + 4 + 2 = 8$ Ans

(+) Why the Best-First search is considered "greedy".

⇒ Greedy best-first search algorithm always selects the ~~path~~ node with minimum heuristic value for the exploration. This algorithm does not take care about the actual path cost. As the heuristic value ($h(n)$) is an estimated value, it may be more than the actual cost sometime.

Thus it prioritizes paths that appear to be the most promising, regardless of whether or not they are actually the shortest path. Therefore, it is called greedy. It does not always guaranteed to find the optimal solution.

Artificial Intelligence (AI)
CS-3011
Evaluation Scheme for the Mid-Semester Examination (Autumn 2023)

Q3. (a) Illustrate the significance of PEAS in Artificial Intelligence. Specify the following agents based on PEAS to determine their task environments in a tabular manner:
I. Chandrayaan-3 Vikram lander
II. Robotic news anchor

PEAS (Performance, Environment, Actuators, Sensors) gives a description of the task environment in which the agent is supposed to function. These specifications define the complexity of the problem and the kind of agent programming that is required for the agent.

I.Chandrayaan-3 Vikram lander
Performance measure – Accuracy of landing, location, transmission of signals, distance covered, responsiveness, connection with rover.
Environment – Moon surface, rocks, craters, temperature, dust, lighting.
Actuators – Legs, thrusters, deployers, scientific instruments.
Sensors – Cameras, altimeters, hazard detector, temperature sensor, spectrometer.

II.Robotic news anchor
Performance measure – Accuracy, clarity, comprehensiveness, intonation, timeliness.
Environment – Newsroom, studio, live feed, prompts.
Actuators – Speaker, gesture generator, voice modulator.
Sensors – Camera, sonar / microphone.
[Marks 1 + 2 + 2, step marking as per faculty discretion]

Q3. (b) Define and design the state-space graph for Missionaries and Cannibals problem in Artificial Intelligence. How can various search algorithms, such as Breadth-first search, and Depth-first search, be applied to explore the state-space graph of the Missionaries and Cannibals problem, and what are the trade-offs in terms of efficiency and optimality among these algorithms?

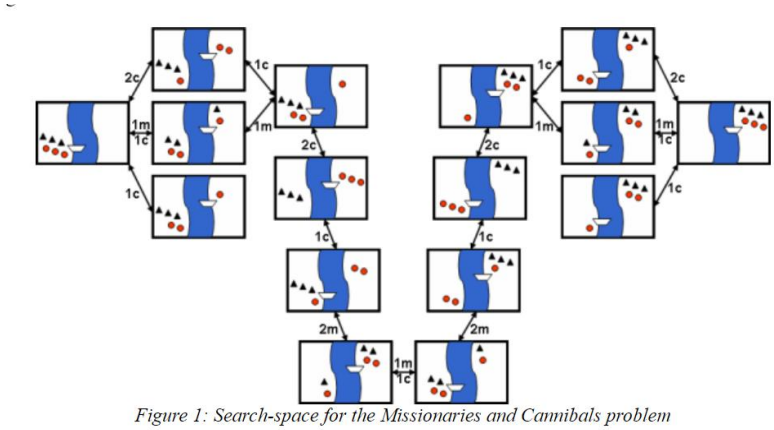


Figure 1: Search-space for the Missionaries and Cannibals problem

[Marks 5, step marking as per faculty discretion]

Artificial Intelligence (AI)
CS-3011
Evaluation Scheme for the Mid-Semester Examination (Autumn 2023)

1. Missionaries and Cannibals.

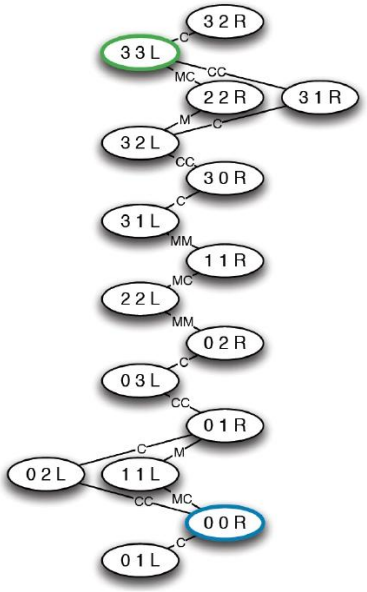
Missionaries and Cannibals is a problem in which 3 missionaries and 3 cannibals want to cross from the left bank of a river to the right bank of the river. There is a boat on the left bank, but it only carries at most two people at a time (and can never cross with zero people). If cannibals ever outnumber missionaries on either bank, the cannibals will eat the missionaries.

A state can be represented by a triple, (*m c b*), where *m* is the number of missionaries on the left, *c* is the number of cannibals on the left, and *b* indicates whether the boat is on the left bank or right bank. For example, the initial state is (3 3 L) and the goal state is (0 0 R).

Operators are:

- MM: 2 missionaries cross the river
- CC: 2 cannibals cross the river
- MC: 1 missionary and 1 cannibal cross the river
- M: 1 missionary crosses the river
- C: 1 cannibal crosses the river

Draw a diagram showing all the legal states and transitions from states corresponding to all **legal** operations. See Figure 3.3 in Russell & Norvig (p. 65) for an example of what your diagram should look like.



2. Answer the following question.

Hint: think about branching factor of states near the initial state and goal state, as compared to the branching factor of states in the "middle" of the state space.

3. Breadth-first search

Solve the Missionaries and Cannibals problem by implementing the BFS algorithm given in class. Implement the BFS algorithm to show the open list and closed list at every iteration of the algorithm until the goal is visited. Use the format below. I have given the first two iterations.

open: 33L
closed: nil

open: 31R, 22R, 32R
closed: 33L

open: 22R, 32R, 32L
closed: 31R, 33L

---

open: 32R, 32L, 32L
closed: 22R, 31R, 33L

open: 32L, 32L
closed: 32R, 22R, 31R, 33L

open: 32L, 30R
closed: 32L, 32R, 22R, 31R, 33L

open: 30R
closed: 32L, 32R, 22R, 31R, 33L

open: 31L
closed: 30R, 32L, 32R, 22R, 31R, 33L

open: 11R
closed: 31L, 30R, 32L, 32R, 22R, 31R, 33L

open: 22L
closed: 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L

open: 02R
closed: 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L

open: 03L
closed: 02R, 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L

open: 03L
closed: 11R, 02R, 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L

open: 01R
closed: 03L, 11R, 02R, 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L

open: 11L, 02L
closed: 01R, 03L, 11R, 02R, 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L

open: 02L, 00R, 01R
closed: 11L, 01R, 03L, 11R, 02R, 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L

open: 00R, 01R, 00R,
closed: 02L, 11L, 01R, 03L, 11R, 02R, 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L

Solution: CC, C, CC, C, MM, MC, MM, C, CC, M, MC

---

3. Depth-first search

Same as problem 2, but use the **DFS** algorithm given in class.

open: 33L
closed: nil

open: 31R, 22R, 32R
closed: 33L

open: 32L, 22R, 32R
closed: 31R, 33L

open: 30R, 22R, 32R
closed: 32L, 31R, 33L

open: 31L, 22R, 32R
closed: 30R, 32L, 31R, 33L

open: 11R, 30R, 22R, 32R
closed: 31L, 30R, 32L, 31R, 33L

open: 22L, 30R, 22R, 32R
closed: 11R, 31L, 30R, 32L, 31R, 33L

open: 02R, 30R, 22R, 32R
closed: 22L, 11R, 31L, 30R, 32L, 31R, 33L

open: 03L, 30R, 22R, 32R
closed: 02R, 22L, 11R, 31L, 30R, 32L, 31R, 33L

open: 01R, 30R, 22R, 32R
closed: 03L, 02R, 22L, 11R, 31L, 30R, 32L, 31R, 33L

open: 11L, 02L, 30R, 22R, 32R
closed: 01R, 03L, 02R, 22L, 11R, 31L, 30R, 32L, 31R, 33L

open: 00R, 01R, 02L, 30R, 22R, 32R
closed: 11L, 01R, 03L, 02R, 22L, 11R, 31L, 30R, 32L, 31R, 33L

Solution: CC, C, CC, C, MM, MC, MM, C, CC, M, MC

Or

Artificial Intelligence (AI)
CS-3011
Evaluation Scheme for the Mid-Semester Examination (Autumn 2023)

Breadth-First-Search

open: 33L	closed: nil
open: 31R, 22R, 32R	closed: 33L
open: 22R, 32R, 32L	closed: 31R, 33L
open: 32R, 32L, 32L	closed: 22R, 31R, 33L
open: 32L, 32L	closed: 32R, 22R, 31R, 33L
open: 32L, 30R	closed: 32L, 32R, 22R, 31R, 33L
open: 30R	closed: 32L, 32R, 22R, 31R, 33L
open: 31L	closed: 30R, 32L, 32R, 22R, 31R, 33L
open: 11R	closed: 31L, 30R, 32L, 32R, 22R, 31R, 33L
open: 22L	closed: 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L
open: 02R	closed: 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L
open: 03L	closed: 02R, 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L
open: 03L	closed: 11R, 02R, 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L
open: 01R	closed: 03L, 11R, 02R, 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L
open: 11L, 02L	closed: 01R, 03L, 11R, 02R, 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L
open: 02L, 00R, 01R	closed: 11L, 01R, 03L, 11R, 02R, 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R, 31R, 33L
open: 00R, 01R, 00R, 31R, 33L	closed: 02L, 11L, 01R, 03L, 11R, 02R, 22L, 30R, 11R, 31L, 30R, 32L, 32R, 22R,

Solution: CC, C, CC, C, MM, MC, MM, C, CC, M, MC

Depth-First-Search

open: 33L	closed: nil
open: 31R, 22R, 32R	closed: 33L
open: 32L, 22R, 32R	closed: 31R, 33L
open: 30R, 22R, 32R	closed: 32L, 31R, 33L
open: 31L, 22R, 32R	closed: 30R, 32L, 31R, 33L
open: 11R, 30R, 22R, 32R	closed: 31L, 30R, 32L, 31R, 33L
open: 22L, 30R, 22R, 32R	closed: 11R, 31L, 30R, 32L, 31R, 33L
open: 02R, 30R, 22R, 32R	closed: 22L, 11R, 31L, 30R, 32L, 31R, 33L
open: 03L, 30R, 22R, 32R	closed: 02R, 22L, 11R, 31L, 30R, 32L, 31R, 33L
open: 01R, 30R, 22R, 32R	closed: 03L, 02R, 22L, 11R, 31L, 30R, 32L, 31R, 33L
open: 11L, 02L, 30R, 22R, 32R	closed: 01R, 03L, 02R, 22L, 11R, 31L, 30R, 32L, 31R, 33L
open: 00R, 01R, 02L, 30R, 22R, 32R	closed: 11L, 01R, 03L, 02R, 22L, 11R, 31L, 30R, 32L, 31R, 33L

Solution: CC, C, CC, C, MM, MC, MM, C, CC, M, M

Q4. (a) Define the “state-space” in Artificial Intelligence. How does one stop the repeated states in a state-space search?

In artificial intelligence, the "state space" refers to the set of all possible states that a problem-solving agent or algorithm can encounter while trying to reach a goal or find a solution to a problem. State spaces are commonly used in various AI problem-solving approaches, including search algorithms, planning, and optimization.

Here's a more detailed explanation:

State: A state represents a specific configuration or situation of a problem at a particular point in time. In state-space search, each node in the search tree or graph corresponds to a state.

Initial State: This is the starting point of the problem, representing the initial configuration or situation.

Goal State: The goal state represents the desired outcome or solution to the problem. The objective of the AI agent or algorithm is to find a sequence of actions or transitions from the initial state to a goal state.

Operators/Actions: Operators or actions are the allowable moves, transitions, or transformations that can be applied to move from one state to another. These actions define the edges in the state-space graph.

Artificial Intelligence (AI)
CS-3011
Evaluation Scheme for the Mid-Semester Examination (Autumn 2023)

Path: A path is a sequence of states and actions that lead from the initial state to a goal state.
State-Space Search: State-space search algorithms explore this graph or tree of states and actions to find a path from the initial state to the goal state. Common search strategies include breadth-first search (BFS), depth-first search (DFS), A* search, and others.

To prevent repeated states in a state-space search, which can lead to inefficiency and infinite loops, you can use several techniques:
Closed List or Visited Nodes: Maintain a data structure, often called a "closed list" or "visited nodes," to keep track of states that have already been explored. Before expanding a new state, check if it is in the closed list. If it is, you can skip exploring it again.
State Hashing: Assign a unique identifier (hash value) to each state. Store these identifiers in a data structure like a hash table or a set. Before expanding a new state, check if its hash value is already in the set. If it is, you've already explored that state.
Cycle Detection: Implement cycle or loop detection mechanisms in your search algorithm. For example, in depth-first search (DFS), you can check for cycles by keeping track of the path you've traversed and ensuring you don't revisit a state that's already on the current path.
Graph Search vs. Tree Search: In some cases, you might choose to use a graph search instead of a tree search. In a graph search, you ensure that each state is expanded only once, even if it's reachable through multiple paths. Tree search, on the other hand, explores the same state multiple times if it's encountered through different branches of the search tree.

By implementing one or more of these techniques, you can prevent the repeated exploration of states in a state-space search, which helps improve the efficiency and correctness of your AI problem-solving algorithm.
[Marks 2 + 2, step marking as per faculty discretion]

Q4. (b) A Mars rover has to leave the lander, collect rock samples from three places (in any order) and return to the lander. Assume that it has a navigation module that can take it directly from any place of interest to any other place of interest. So it has primitive actions **go-to-lander, go-to-rock-1, go-to-rock2, and go-to-rock3**.

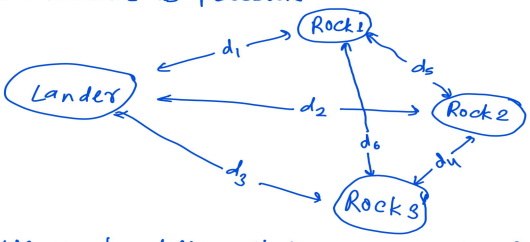
We know the time it takes to traverse between each pair of special locations. Our goal is to find a sequence of actions that will perform this task in the shortest amount of time.
I.Formulate this problem as a search problem by specifying the state space, initial state, actions, transition model, path-cost function, and goal test.
II.Explain which search technique would be most appropriate to solve this problem, and why?

[Marks 4 + 2, step marking as per faculty discretion]

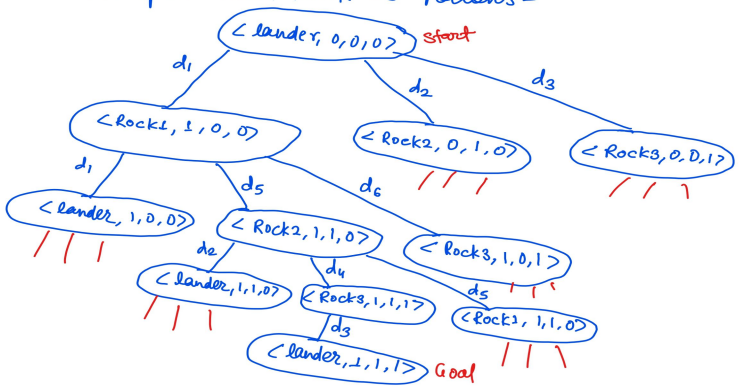
We can define state space for the problem in following form where each state will look like -
< Location, hasRock1?, hasRock2?, hasRock3? > where
Location $\in$ { Lander, Rock1, Rock2, Rock3 }
hasRocki $\in$ { True, False }
⊥ 0
So Initial state would look like < lander, 0,0,0 >
and Goal state would look like < lander, ⊥, ⊥, ⊥ >
and set of actions is go-to-lander, go to rock1, etc.
which is given in question description itself.

Artificial Intelligence (AI)
CS-3011
Evaluation Scheme for the Mid-Semester Examination (Autumn 2023)

To define path cost, let's assume some distances between rocks and lander as follows –



Now we can treat these distances as cost of one action and define path cost as sum of arc costs.
A part of state space could look as follows –



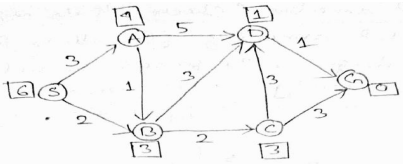
This is a part of state space, entire state space will have more states.
Now that we have defined state space, our goal is to find path from start state to goal state for which many search algorithms can be used like BFS, DFS, etc. But since path cost is given so we can also use UCS for finding shortest path. Therefore UCS will be best choice available.

Q5. Consider the graph given below. Assume that S is the initial state and G is the goal state. The transition costs (or step costs) are next to the edges, and the heuristic values are indicated within round brackets very close to the states.

Compare and apply
i) UCS, and
ii) A* algorithm to find the path and path-cost in each case. (Draw in each case the tree diagram.)

[Marks 5 for UCS + 5 for A*, step marking as per faculty discretion]

Artificial Intelligence (AI)
CS-3011
Evaluation Scheme for the Mid-Semester Examination (Autumn 2023)



UCS :- Initial state : S
Goal state : G
$f(n) = g(n)$ [only cost of edge]

Priority Queue	Explored nodes
S^0	S
$B^2 A^3$	S, B
$A^3 C^4 D^5$	S, B, A
$C^4 D^5 D^8$	S, B, A, C
$D^5 D^7 G^7 D^8$	S, B, A, C, D
$G^6 G^7$	

as D is already explored in a lower cost path we discard D^7, D^8

The lowest cumulative path cost is G^6.

output :- $S \xrightarrow{2} B \xrightarrow{3} D \xrightarrow{1} G$
path - cost = 6.

A* Algo :- Initial state : S
Goal state : G
$f(n) = g(n) + h(n)$
↑ path cost ↑ heuristic value

Explored nodes

Iteration 1 :- $f(S \to A) = 3 + 4 = 7$ [S]
$f(S \to B) = 2 + 3 = 5$ ✓

Iteration 2 :- $f(S \to A) = 7$ [S, B]
$f(S \to B \to C) = 4 + 3 = 7$
$f(S \to B \to D) = 5 + 1 = 6$ ✓

Iteration 3 :- $f(S \to A) = 7$ [S, B, D]
$f(S \to B \to C) = 7$
$f(S \to B \to D \to G) = 6 + 0 = 6$ ✓
Reached the goal node

output :- $S \to B \to D \to G$
Path cost = 6.