USN | | | | | | | | | |

CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A++ GRADE BY NAAC

Internal Assessment Test 2 – February2024

Sub:	**Principles of Artificial Intelligence**					Sub Code:	**21AI54**	Branch:	**AIML**	
Date:	**06/02/24**	Duration:	**90 minutes**	Max Marks:	**50**	Sem/Sec:	**V - A**		**OBE**	
Answer any FIVE FULL Questions								**MARKS**	**CO**	**RBT**

USN

			[6]		

1a) Compare depth first search, iterative deepening depth first search and bi-directional search. (3 x 2=6)

The properties of depth-first search depend strongly on whether the graph Search or tree-search version is used. The graph search version is complete, where as tree search version is not complete.

Both graph and tree versions are not optimal. Depth first tree search may generate all of $O(b^m)$ nodes where m is the maximum depth of any node

L2

For a state space with branching factor b and maximum depth m, the depth first search requires storage of O(bm) nodes.

DFS fails in infinite state space.

Iterative deepening DFS

Iterative deepening DFS finds the best depth limit. By gradually increasing the limit -0,1,2……. until a goal is found – when depth d is reached. The space complexity is O(bd)

2

The time complexity is $O(b^d)$ - same as breadth first search

Bidirectional Search

One forward search from initial state to goal state and one backward search from goal state to initial state. The motivation is $b^{d/2}+b^{d/2}$ is much less than $O(b^{d/2})$. Bidirectional Search is complete. Space complexity is $O(b^d)$. The search is optimal as well.

1b) State whether the following statement is true or false: 'Iterative deepening depth first search has same asymptotic space complexity as breadth first search'. Justify your answer. (4)

L3

In iterative deepening states are generated multiple times. But in a search tree with same branching factor, most nodes at the bottom levels. Nodes at depth d are generated once and that at d-1 are generated twice and so on and children of root are generated d times. Number of nodes generated = (d) b+ (d-1) b^2+…….+ (1) $b^d=O(b^d)$, same as that of BFS.

USN | | | | | | | | | |

		2a) Explain A Search and the conditions for optimality of informed search strategies.(3+3=6)* A* is a best-first heuristic search. The heuristic function is defined as f(n) =g(n0 + h(n) where g(n) is the cost to search node and h(n) being the cost to reach from the node n to goal state. The condition for optimality are admissibility and consistency. h(n) should be in admissible heuristic - An Admissibility heuristic is one that never overestimates the cost to reach the goal. A heuristic h(n) is consistent, if for every nodes n and every node successor node n' of n, generated by 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'. h(n) <= c(n, a, n') + h(n)') This is a form of general triangular inequality. *2b) Prove that that A* search is optimal.(2+2=4)* Proof of optimality of A*: Tree search version of A* is optimal if h(n) is admissible, which graph search version is optimal, if h(n) is consistent. We establish that h(n) is consistent. If h(n) is consistent the vales of f(n) along any path are non-decreasing. Proof: Suppose n' is a successor of n, then g(n') = g(n) + c(n, a, n') for some action a. f(n') = g(n') + h(n') = g(n) + c(n,a,n') + h(n') $\geq$ g(n)+ h(n)=f(n) The next step is that whenever A* selects a node n for expansion, the optimal path to that node has been found. Assume this is not the case, frontier then there would have to be another node n' on the optimal path from the state node to n, became f' is non decreasing along any path, n' would have lower & cost than n, and would have been selected first. From the preceding observation, it follows that the sequence of nodes expanded using A* using Graph Search is in non-decreasing order of them. hence their first node selected for expansion must be an optimal solution.		2	L2 L3

USN

3) Explain strategies to generate admissible heuristics.(3x2 = 6)

A problem with fewer restrictions on the actions is relaxed problem: State space graph of a relaxed problem is state space is a super graph of the original state space -creates added edges. Therefore any optimal solution in this original problem is by definition also a solution in the relaxed problem. Derived heuristic obeys is triangular inequality and is consistent.

Pattern Database : Admissible heuristic can be derived from the solution cost of subproblem of a given problem. Store these exact solution costs of every possible sub problem instance. Then we compute an admissible heuristic h_{DB} for each complete state encountered during a search simply by looking up the corresponding subproblem configuration in database. The database itself is constructed by going back from the goal and recording the cost of each new pattern encountered.
Learning heuristics from experience.
A heuristic function h(n) is supposed to estimate the cost of the solution beginning from the state at node n. The agent could construct such a function by learning from experience. Each example consists of a state from the solution path and the actual cost of the solution from that point. From these examples, a learning algorithm can be used to construct a function h(n), that can predict solution costs for other states that arise during search.

4a) Explain Hill Climbing Algorithm. What are the drawbacks of Hill Climbing Algorithm? (3+3=6)

Algorithms like A* needs to store one or more paths in the memory.
If a path to goal does not matter, we might consider local search.
Local search algorithms operate using single current a single current node and generally move only to neighbors.

Hill climbing algorithm - Steepest ascent is a loop that continually moves in the direction of increasing value. It terminates when it reaches a peak, where no neighbors has a higher value.

```
function HILL-CLIMBING (problem) return a state that is local max
current <- MAKENODE (problem.INITIAL STATE)
loop do
 neighbor <- a highest valued successor of current
 if neighbor.VALUE <= current.VALUE then return current.STATE
 current <-   neighbor
```

Marks	CO	RBT
[10]	2	L2

Drawback:

1) Get's stuck at local maxima
2) Ridges result in a sequence of local maxima that is very difficult for the greedy algorithm to navigate
3) The hill climbing algorithms will get stuck on a plateau.

4b) *Write pseudo code for Simulated Annealing algorithm. (2)*
Differentiate Simulated Annealing from Hill Climbing. (2)

function SIMULATED - ANNEALING (Problem, schedule) return A Solution state

Inputs:

problem

schedule

current <- MAKE NODE (Problem. INITIAL-STATE)
for i = 1 to infinity do
  T <- schedule(t)
  if T = 0 then return current
  next <- a randomly selected successor of current
  Delta E <- next. VALUE - current VALUE

  if Delta E >0 then current <- next

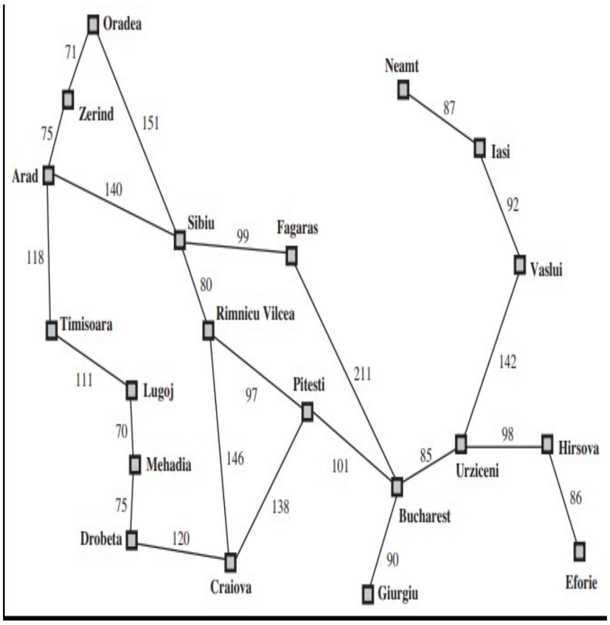
  else current <- next only with probability $e^{\text{Delta E / T}}$

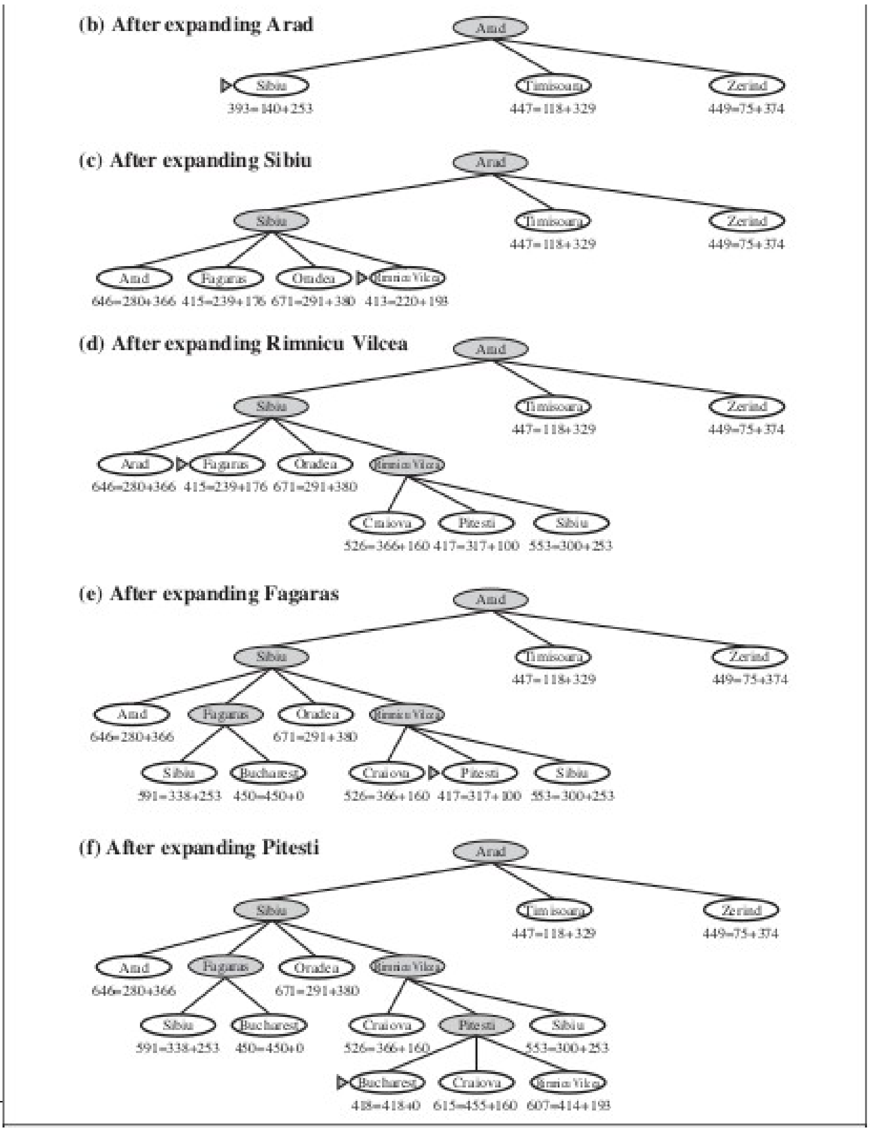
Hill climbing algorithm never moves downward from the hill towards a lower value and therefore is incomplete, because it gets stuck on the local maxima. Simulated annealing allows the movement in either direction.

[10]	2	L3

5) Map of Romania and the straight line distance from various cities to Bucharest is given below. Apply A* search algorithm and determine the optimal Route from Arad to Bucharest.

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374





		6) Define the following terms with respect to logic: (5 x 2 = 10) (i)Relation 'satisfies': Satisfies: If a sentence x is true in m, we say that m *satisfies* x. A model is a mathematical abstraction, each of which fixes the truth or the falsehood of every relevant sentence. (ii)relation 'entailment' : X *entails* B iff in every model in which X is true B is also true. M(X) is the set of all models of x X *entails* B iff M(x) is a sub set of M(B) (iii)Model checking : Model checking is an inference technique in which to establish KB entails x as all possible models are enumerated to check that x is true in all models in which KB is true (iv) Soundness of inference algorithms An inference algorithm that derives only entiled sentences is called sound. (v) Completeness of inference algorithm An inference algorithm is complete if it can drive any sentence that is entitled.	[10]	2	L1