INTELLIGENT AGENT

Multiple Choice Type Questions

[WBUT 2017]

1. Agents are
   a) autonomous
   b) adaptive
   c) both (a) and (b)
   d) none of these

Answer: (c)

Short Answer Type Questions

1. a) What is perception? [WBUT 2008]

Answer:

In perception the environment is scanned by means of various sensory organs, real or artificial, and the scene is decomposed into separate objects in various spatial relationships.

b) Define intelligent agent. [WBUT 2008]

OR,

What is an agent? [WBUT 2011]

Explain different types of environment related to intelligent agent. [WBUT 2008]

OR,

What is an agent? Describe various agent types. [WBUT 2016]

OR,

What is an agent in AI? What are the types of agent? Discuss about environment for agent. [WBUT 2017, 2018]

Answer:

Intelligent Agent:

An **agent** is anything that can be viewed as **perceiving** its environment through **sensors** and **acting** upon that environment through **effectors**. An agent is a computer system situated in some environment, and that is capable of *autonomous action* in this environment in order to meet its design objectives.

An intelligent agent is a computer system that is capable of *flexible* autonomous action in order to meet its design objectives. By *flexible,* we mean that the system must be:

- ***responsive***: agents should perceive their environment (which may be the physical world, a user, a collection of agents, the Internet, etc.) and respond in a timely fashion to changes that occur in it,
- ***proactive***: agents should not simply act in response to their environment, they should be able to exhibit opportunistic, goal-directed behaviour and take the initiative where appropriate, and
- ***social***: agents should be able to interact, when they deem appropriate, with other artificial agents and humans in order to complete their own problem solving and to help others with their activities.

Intelligent agents are usually classified into five classes based on their degree of perceived intelligence and capability:

1. simple reflex agents
2. model-based reflex agents
3. goal-based agents
4. utility-based agents
5. learning agents.

The environment can be classified into the following:

Accessible* vs. *inaccessible

An accessible environment is one in which the agent can obtain complete, accurate, up-to-date information about the environment's state. Most moderately complex environments (including, for example, the everyday physical world and the Internet) are inaccessible. The more accessible an environment is, the simpler it is to build agents to operate in it.

Deterministic* vs. *non-deterministic

As we have already mentioned, a deterministic environment is one in which any action has a single guaranteed effect. The physical world can to all intents and purposes be regarded as non-deterministic. Non-deterministic environments present greater problems for the agent designer.

Episodic* vs. *non-episodic

In an episodic environment, the performance of an agent is dependent on a number of discrete episodes, with no link between the performance of an agent in different scenarios. Episodic environments are simpler from the agent developer's perspective because the agent can decide what action to perform based only on the current episode - it need not reason about the interactions between this and future episodes.

Static* vs. *dynamic

A static environment is one that can be assumed to remain unchanged except by the performance of actions by the agent. A dynamic environment is one that has other processes operating on it, and which hence changes in ways beyond the agent's control. The physical world is a highly dynamic environment.

Discrete* vs. *continuous

An environment is discrete if there are a fixed, finite number of actions and percepts in it. Russell and Norvig give a chess game as an example of a discrete environment, and taxi driving as an example of a continuous one.

2. What are the disadvantages of table driven agent? [WBUT 2011]

Answer:

Disadvantages of Table driven agent

1. The table needed for something as simple as an agent that can only play chess would be about 3510° entries.
2. It would take quite a long time for the designer to build the table.
3. The agent has no autonomy at all, because the calculation of best actions is entirely built in.

So if the environment changed in some unexpected way, the agent would be lost.

4. Even if we gave the agent a learning mechanism as well, so that it could have a degree of autonomy, it would take forever to learn the right value for all the table entries.

3. a) What is percept sequence? **[WBUT 2016]**
b) What is agent system?

Answer:

a) A percept is an input that an intelligent agent receives at a given moment. The **percept sequence** is the complete history of every and any percept that has been received. Importantly, a rational agent's choice of action can only depend on its percept sequence and - conversely - not on anything it hasn't perceived.

b) Multi-agent systems consist of agents and their environment. A multi-agent system is a computerized system composed of multiple interacting intelligent agents within an environment. Multi-agent systems can be used to solve problems that are difficult or impossible for an individual agent or a monolithic system to solve.

Long Answer Type Questions

1. Discuss on 'agents as search procedure'. **[WBUT 2011]**

Answer:

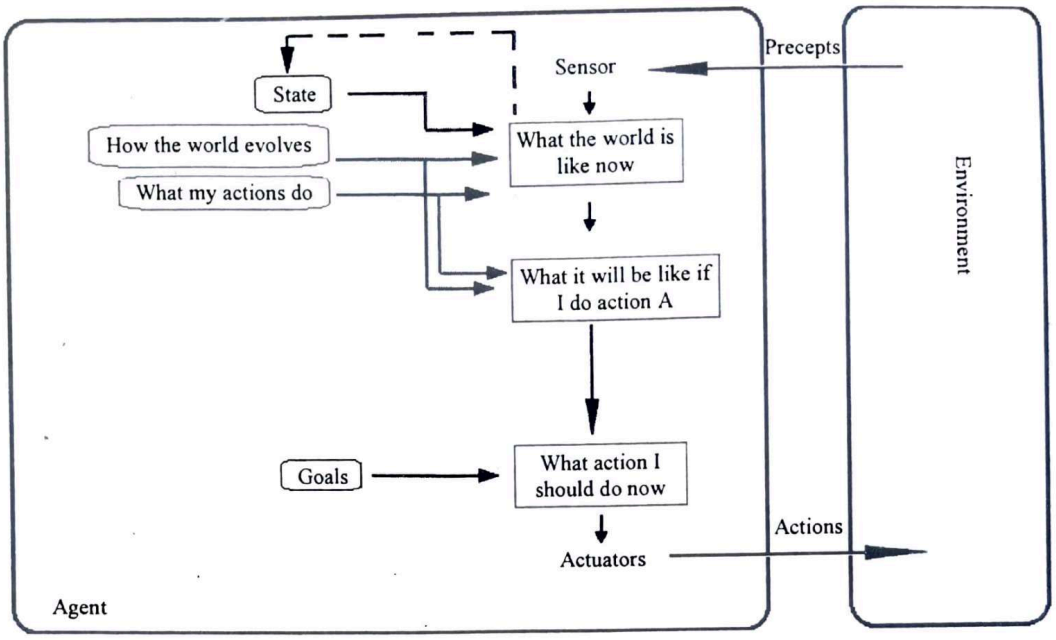
This notion of search is computation inside the agent. It is different from searching in the world, when it may have to act in the world, for example, an agent searching for its keys, lifting up cushions, and so on. It is also different from searching the web, which involves searching for information. Searching in this chapter means searching in an internal representation for a path to a goal.

The idea of search is straightforward: the agent constructs a set of potential partial solutions to a problem that can be checked to see if they truly are solutions or if they could lead to solutions. Search proceeds by repeatedly selecting a partial solution, stopping if it is a path to a goal, and otherwise extending it by one more arc in all possible ways. When an agent is given a problem, it is usually given only a description that lets it recognize a solution, not an algorithm to solve it. It has to search for a solution.

2. a) Describe goal based agent system. **[WBUT 2017]**

Answer:

Goal-based agents expand on the capabilities of the model-based agents, by using "goal" information. Goal information describes situations that are desirable. This allows the agent a way to choose among multiple possibilities, selecting the one which reaches a goal state. Search and planning are the subfields of artificial intelligence devoted to finding action sequences that achieve the agent's goals.



b) What do you mean by a table driven agent? What is the problem of this agent? **[WBUT 2017]**

Answer:

1st part: In table driven agent, the agent uses a lookup table for actions to be taken for every possible state of the environment.

2nd part: ***Refer to Question No. 3(b) of Short Answer Type Questions.***

4. Write short note on Intelligent Agents. **[WBUT 2006, 2013]**

Answer:

Refer to Question No. 1(b) of Short Answer Type Questions.

PROBLEM SOLVING & SEARCHING

Multiple Choice Type Questions

1. Heuristic search has [WBUT 2006, 2013]
   a) minimization of function value
   b) maximization of function value
   c) both (a) & (b)
   d) none of these

Answer: (c)

2. Decomposable problem can be represented by [WBUT 2006, 2007, 2011]
   a) OR graph b) AND graph c) AND-OR graph d) None of these

Answer: (c)

3. Which is not heuristic search? [WBUT 2006, 2008, 2010, 2013]
   a) Constrained satisfaction search
   b) Depth first search
   c) Simulated annealing
   d) Steepest ascent Hill climbing

Answer: (b)

4. Algorithm that gives optimal solution [WBUT 2007, 2009, 2010, 2012, 2013, 2017, 2018]
   a) Hill climbing b) BFS c) Blind search d) A *

Answer: (d)

5. Which of the following is not a conflict resolution strategy in production system? [WBUT 2008, 2014, 2015]
   a) Production rules b) Recency c) Refractoriness d) Specificity

Answer: (a)

6. Uninformed search is also known as [WBUT 2010, 2017]
   a) Brute force search
   b) Hill climbing search
   c) Worst case search
   d) Blind search

Answer: (d)

7. The time complexity and space complexity for bidirectional breadth first search technique, respectively are (with branching factor b and depth d) [WBUT 2011]
   a) $O(b^d), O(b^d)$
   b) $O(b^{\frac{d}{2}}), O(b^{\frac{d}{2}})$
   c) $O(b^{\frac{d}{2}}), O(b^d)$
   d) $O(b^d), O(b^{\frac{d}{2}})$

Answer: (b)

8. The term 'Optimality', so far one of the performance measuring indices of any search technique is concerned, refers to [WBUT 2011]
   a) time complexity
   b) space complexity
   c) both (a) and (b)
   d) none of these

Answer: (c)

9. The main advantage of any heuristic search algorithm over blind search one is with respect to [WBUT 2011]
a) time complexity b) space complexity
c) completeness d) optimality

Answer: (b)

10. Depth first search procedure uses [WBUT 2011, 2017]
a) AND graph b) OR graph c) AND-OR graph d) none of these

Answer: (d)

11. In Minimax algorithm search process obeys [WBUT 2011, 2017, 2018]
a) breadth first search fashion b) depth first search fashion
c) best first search fashion d) none of these

Answer: (b)

12. Iterative Deepening search procedure is [WBUT 2011]
a) optimal with respect to time consumption
b) optimal with respect to space consumption
c) both (a) and (b)
d) none of these

Answer: (c)

13. Searching techniques are used for [WBUT 2012]
a) goal node searching
b) optimization of search space
c) finding goal distance of the goal node from start node
d) all of these

Answer: (d)

14. Hill climbing has potential problems like [WBUT 2012]
a) lake b) foothill trap c) garden d) all of these

Answer: (b)

15. The form of heuristic function of A* is [WBUT 2012]
a) $f^*(n)=g^*(n)*h^*(n)$ b) $f^*(n)=g^*(n)+h^*(n)$
c) $f^*(n)=g^*(n)+h(n)$ d) none of these

Answer: (d)

16. Inheritable knowledge is best presented by [WBUT 2012]
a) OR graph b) AND graph c) AND-OR graph d) none of these

Answer: (d)

17. If in a problem the number of initial states is much more than the number of final states we should use [WBUT 2014, 2015]
a) backward reasoning b) forward reasoning
c) both (A) and (B) d) none of these

Answer: (a)

[WBUT 2015]

18. Which is NOT a heuristic search?
    a) A* search
    b) steepest ascent Hill-climbing
    c) Simulated annealing
    d) Depth first search

Answer: (d)

[WBUT 2016]

19. What is meant by simulated annealing in artificial intelligence?
    a) Returns an optimal solution when there is a proper cooling schedule
    b) Returns an optimal solution when there is no proper cooling schedule
    c) It will not return an optimal solution when there is a proper cooling schedule
    d) None of the mentioned

Answer: (a)

[WBUT 2016]

20. A* algorithm is based on
    a) Breadth-first-search
    b) Depth-first-search
    c) Best-first-search
    d) Hill climbing

Answer: (c)

21. Which search agent operates by interleaving computation and action?
    a) Offline search
    b) Online search
    c) Breadth-first search
    d) Depth-first search

[WBUT 2016]

Answer: (b)

22. Which search method takes less memory?
    a) Depth-First Search
    b) Breadth-First search
    c) Both (a) and (b)
    d) Linear Search

[WBUT 2016]

Answer: (a)

23. Which is the most straight forward approach for planning algorithm?
    a) Best-first search
    b) State-space search
    c) Depth-first search
    d) Hill-climbing search

[WBUT 2016]

Answer: (b)

24. Uniformed search is also known as:
    a) Brute force search
    b) Hill climbing search
    c) Worst case search
    d) Blind search

[WBUT 2018]

Answer: (d)

25. Depth first search producers uses
    a) AND graph
    b) OR graph
    c) AND-OR graph
    d) None of these

[WBUT 2018]

Answer: (d)

Short Answer Type Questions

1. Discuss the benefits of a production system. **[WBUT 2006, 2010, 2012]**

OR,

What is a production system? **[WBUT 2016]**

Answer:

A production system is a method of modeling human problem solving. When trying to solve everyday problems, people may respond to an external condition with appropriate actions; they also form plans to guide their behavior. One way to uncover a person's problem-solving strategies is to collect a protocol: the problem solver's own account of thoughts and intentions, spoken while solving the problem. From these `think aloud' protocols, the researcher tries to construct a problem space consisting of the person's states of knowledge and the operators needed to move from one state to another. A production system is one means of encoding a problem space, in a form that can be run on a computer. It has three main components: a database, called the working memory that represents a person's current state of knowledge; a set of production rules that operate on the working memory; and an interpreter that decides which rule to fire in a particular state.

Production systems have certain advantages over conventional computer programs that make them suitable for use in expert systems: programs that simulate the performance of human experts.

The benefits are:

- **Flexibility:** Production systems use the same basic IF-THEN format to represent knowledge in very different domains.
- **Modularity:** The functional units of a production system are independent, self-contained chunks of knowledge, any one of which can be altered or replaced without disabling the entire production system and without requiring the modification of other rules. Such alterations might modify or restrict the behavior of the system, but will not cripple it. This is because the rules in a production system are separate from the program that runs them: the rules do not interact with one another directly but only through changes to the working memory.
- **Cognitive plausibility:** Production systems can be made to reason either forwards, from initial evidence towards a conclusion, or backwards, from a hypothesis to the uncovering of the right kind of evidence that would support that hypothesis, or by a combination of the two.

2. What is combinatorial explosion? **[WBUT 2007]**

OR,

Briefly discuss combinatorial explosion. **[WBUT 2016, 2017]**

Answer:

The naive way of solving combinatorial problems can be paraphrased as `generate and test': In a first step one enumerates all combinations from which one selects all solutions in the second step. In most cases however, `generate and test' is simply not feasible. This is obvious if the set of combinations is infinite. But even if it is finite then it is usually

very large, i.e. exponentially large in size of the problem description. In this case, the generation step runs into a combinatorial explosion (from which it usually returns only several billions of years later).

3. What is the hill-climbing technique? Describe it. **[WBUT 2007]**

OR,

Briefly explain the problems with Hill Climbing search. How it is different from Gradient Descent Search? **[WBUT 2014]**

Answer:

Hill climbing is an optimization technique which belongs to the family of local search. It is a relatively simple technique to implement, making it a popular first choice. Although more advanced algorithms may give better results, there are situations where hill climbing works well. Hill climbing attempts to maximize (or minimize) a function $f(x)$, where x are discrete states. These states are typically represented by vertices in a graph, where edges in the graph encode nearness or similarity of a graph. Hill climbing will follow the graph from vertex to vertex, always locally increasing (or decreasing) the value of f, until a local maximum (or local minimum) x_m is reached. Hill climbing can also operate on a continuous space: in that case, the algorithm is called gradient ascent (or gradient descent if the function is minimized).

The algorithm is started with a random (potentially bad) solution to the problem. It sequentially makes small changes to the solution, each time improving it a little bit. At some point the algorithm arrives at a point where it cannot see any improvement anymore, at which point the algorithm terminates. Ideally, at that point a solution is found that is close to optimal, but it is not guaranteed that hill climbing will ever come close to the optimal solution.

Gradient descent methods can move in any direction that the ridge or alley may ascend or descend. Hence, gradient descent is generally preferred over hill climbing when the target function is differentiable. Hill climbers, however, have the advantage of not requiring the target function to be differentiable, so hill climbers may be preferred when the target function is complex.

4. What do you mean by completeness of a search? Why DFS is not always complete? **[WBUT 2008, 2012, 2015]**

OR,

What do you mean by completeness of a search method? When do you think BFS & DFS can be incomplete? **[WBUT 2013, 2014]**

Answer:

A search strategy is said to be *complete* if it is guaranteed to find a solution when there is one.

DFS progresses by expanding the first child node of the search tree that appears and thus going deeper and deeper until a goal node is found, or until it hits a node that has no children. Then the search backtracks, returning to the most recent node it hasn't finished exploring. No matter how deep the current node is, DFS will always go deeper if it has a child.

The major weakness of DFS is that it will fail to terminate if there is an infinite path "to the left of" the path to the first solution. In other words, for many problems DFS is not complete: a solution exists but DFS cannot find it.
Both Depth First Search and Breadth First Search are complete for finite state spaces. Both are systematic. They will explore the entire search space before reporting failure. This is because the termination criterion for both is the same. Either they pick the goal node and report success, or they report failure when OPEN (we assume a data structure called OPEN to store the candidates that we have generated.) becomes empty. The only difference is where the new nodes are placed in the OPEN list. Since for every node examined, all unseen successors are put in OPEN, both searches will end up looking at all reachable nodes before reporting failure. If the state space is infinite, but with finite branching then depth first search may go down an infinite path and not terminate. Breadth First Search, however, will find a solution, if there exists one. If there is no solution, both algorithms will not terminate for infinite state spaces.

5. What are the pitfalls of hill-climbing algorithm? **[WBUT 2008, 2014]**

Answer:
Problems of Hill Climbing includes local maxima, ridges and plateau.
Local maxima: A problem with hill climbing is that it will find only local maxima. Unless the heuristic is convex, it may not reach a global maximum.
Ridges: A ridge is a curve in the search place that leads to a maximum, but the orientation of the ridge compared to the available moves that are used to climb is such that each move will lead to a smaller point. In other words, each point on a ridge looks to the algorithm like a local maximum, even though the point is part of a curve leading to a better optimum.
Plateau: Another problem with hill climbing is that of a plateau, which occurs when we get to a "flat" part of the search space, i.e. we have a path where the heuristics are all very close together. This kind of flatness can cause the algorithm to cease progress and wander aimlessly.

6. Compare blind search and heuristic search. **[WBUT 2011]**

Answer:
Sometimes we may not get much relevant information to solve a problem. Suppose we lost our car key and we are not able to recall where we left, we have to search for the key with some information such as in which places we used to place it. It may be our pant pocket or may be the table drawer. If it is not there then we have to search the whole house to get it. The best solution would be to search in the places from the table to the wardrobe. Here we need to search blindly with fewer clues. This type of search is called uninformed search or **blind search**. There are two popular AI search techniques in this category: breadth first search and depth first search.
We can solve the problem in an efficient manner if we have relevant information, clues or hints. The clues that help solve the problem constitute heuristic information. So informed search is called **heuristic search**. Instead of searching one path or many paths just like that informed search uses the given heuristic information to decide whether or not to

explore the current state further. Hill climbing is an AI search algorithm that explores the neighboring states and chooses the most promising state as successor and continue searching for the subsequent states. Once a state is explored, hill climbing algorithm simply discards it. Hill climbing search technique can make substantial savings if it has reliable information. It has to face three challenges: foothill, ridge and plateau. Best first search is a heuristic search technique that stores the explored states as well so that it can backtrack if it realizes that the present path proves unworthy.

7. Discuss on the components of AI production system. **[WBUT 2011]**

Answer:

The major components of an AI production system are

i) A global database
ii) A set of production rules and
iii) A control system

The goal database is the central data structure used by an AI production system.
The production rules operate on the global database. Each rule has a precondition that is either satisfied or not by the database. If the precondition is satisfied, the rule can be applied. Application of the rule changes the database.
The control system chooses which applicable rule should be applied and ceases computation when a termination condition on the database is satisfied. If several rules are to fire at the same time, the control system resolves the conflicts.

8. Did depth limited search always show the completeness property? Explain. **[WBUT 2013, 2014]**

Answer:

Even though depth-limited search cannot follow infinitely long paths, nor can it get stuck in cycles, in general the algorithm is not complete since it does not find any solution that lies beyond the given search depth. But if the maximum search depth is chosen to be greater than the depth of a solution the algorithm becomes complete.

9. Define α-cutoff & β-cutoff of a game tree. **[WBUT 2013, 2014]**

Answer:

When we traverse the search tree in depth-first order:

- At each MAX node n, alpha(n) = maximum value found so far
- At each MIN node n, beta(n) = minimum value found so far

Alpha cutoff: Stop searching below MIN node n if beta(n) <= alpha(i) for some MAX node ancestor i of n.

Beta cutoff: Given a MAX node n, cut off the search below n (i.e., don't generate or examine any more of n's children) if alpha(n) >= beta(i) for some MIN node ancestor i of n.

10. Explain the phenomenon 'Stuck at local optimum' from the perspective of Hill climbing. How can it be resolved using Simulated Annealing? **[WBUT 2013]**

Answer:

Hill climbing attempts to iteratively improve the current state by means of an evaluation function. It always attempts to make changes that improve the current state. Hill-climbing can only advance if there is a higher point in the adjacent landscape. The main problem that hill climbing can encounter is that of "stuck at local optima". This occurs when the algorithm stops making progress towards an optimal solution; mainly due to the lack of immediate improvement in adjacent states.

Simulated annealing solves the problem of "stuck at local minima or maxima" by allowing worse moves (lesser quality) to be taken some of the time. That is, it allows some uphill steps so that it can escape from local minima.

Unlike hill climbing, simulated annealing chooses a random move from the neighbourhood (hill climbing chooses the best move from all those available – at least when using steepest descent (or ascent)).If the move is better than its current position then simulated annealing will always take it. If the move is worse (i.e. lesser quality) then it will be accepted based on some probability.

The law of thermodynamics state that at temperature, t, the probability of an increase in energy of magnitude, δE, is given by

$$P(\delta E) = \exp(-\delta / kt)$$

where k is a constant known as Boltzmann's constant.

The simulation in the algorithm calculates the new energy of the system. If the energy has decreased then the system moves to this state. If the energy has increased then the new state is accepted using the probability returned by the above formula. A certain number of iterations are carried out at each temperature and then the temperature is decreased. This is repeated until the system freezes into a steady state.

This equation is directly used in simulated annealing, although it is usual to drop the Boltzmann constant as this was only introduced into the equation to cope with different materials. Therefore, the probability of accepting a worse state is given by the equation

$$P = \exp(-c/t) > r$$

where

c = the change in the evaluation function
t = the current temperature
r = a random number between 0 and 1

The probability of accepting a worse move is a function of both the temperature of the system and of the change in the cost function. It can be appreciated that as the temperature of the system decreases the probability of accepting a worse move is decreased. This is the same as gradually moving to a frozen state in physical annealing. Also note, that if the temperature is zero then only better moves will be accepted which effectively makes simulated annealing act like hill climbing.

11. Justify the statement:
DFS can be viewed as a special case of Depth-limited search. **[WBUT 2014]**

Answer:

Depth-first search will not find a goal if it searches down a path that has infinite length. So, in general, depth-first search is not guaranteed to find a solution, so it is not complete. This problem is eliminated by limiting the depth of the search to some value l. However, this introduces another way of preventing depth-first search from finding the goal: if the goal is deeper than l it will not be found Regular depth-first search is a special case, for which $l = \infty$.

12. a) What is the difference between Greedy best-first search and A* search?
b) Under what condition is breadth-first search optimal?
c) Show that any monotonic heuristic is admissible. **[WBUT 2015]**

Answer:

a) A* is like Greedy Best-First-Search in that it can use a heuristic to guide itself. In the simple case, it is as fast as Greedy Best-First-Search. However, A* is better as it combines the pieces of information that Dijkstra's algorithm uses (favoring vertices that are close to the starting point) *and* information that Greedy Best-First-Search uses (favoring vertices that are close to the goal).

b) Breadth-first search is optimal if the path cost is a non-decreasing function of the depth of the node. The most common such scenario is that all actions have the same cost.

c) A heuristic function is said to be consistent, or monotone, if it is always at most equal to the estimated distance from any neighboring vertex plus the step cost of reaching that neighbor.

Formally, for every node N and each successor P of N, the estimated cost of reaching the goal from N is no greater than the step cost of getting to P plus the estimated cost of reaching the goal from P. That is: $H(n) \le C(N,P) + h(P)$ and $h(G) = 0$ where

- h is the consistent heuristic function
- N is any node in the graph
- P is any descendant of N
- G is any goal node
- c(N,P) is the cost of reaching node P from N

A consistent heuristic is also admissible, i.e. it never overestimates the cost of reaching the goal (the opposite however is not always true!). This is proved by induction on m, the length of the best path from node to goal. By assumption, $h(N_m) \le h^*(N_m)$, where $h^*(n)$ denotes the cost of the shortest path from n to the goal. Therefore,

$$h(N_{m+1}) \le c(N_{m+1}, N_m) + h(N_m) \le c(N_{m+1}, N_m) + h^*(N_m) = h^*(N_{m+1}),$$

making it admissible. (N_{m+1} is any node whose best path to the goal, of length m+1, goes through some immediate child N_m whose best path to the goal is of length m.)

13. When does BFS give optimal solution? **[WBUT 2016]**

Answer:
Breadth-first search is optimal if the path cost is a non-decreasing function of the depth of the node. The most common such scenario is that all actions have the same cost.

14. What is blind search technique? Explain with examples. [WBUT 2017, 2018]
Answer:
Refer to Question No. 6 (1^{st} part) of Short Answer Type Questions.

15. Show that if a heuristic is consistent then f(n) is monotonically non decreasing along any path. [WBUT 2017]
Answer:
A heuristic is consistent if for every node n, every successor n' of n generated by any action a,

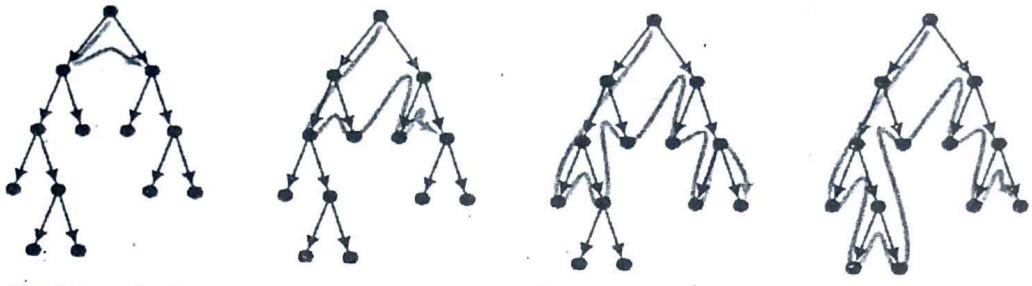
$h(n) \leq c(n,a,n') + h(n')$
If h is consistent, we have
$f(n') = g(n') + h(n')$
$= g(n) + c(n,a,n') + h(n')$
$\geq g(n) + h(n)$
$= f(n)$
i.e., f(n) is non-decreasing along any path.

Hence, if h(n) is consistent, f along any path is non-decreasing.

16. Write iterative deepening algorithm with example. [WBUT 2018]
Answer:
Iterative deepening search calls DFS for different depths starting from an initial value. In every call, DFS is restricted from going beyond given depth. So basically, we do DFS in a BFS fashion. First do DFS to depth 0 (i.e., treat start node as having no successors), then, if no solution found, do DFS to depth 1, etc.

Algorithm
Begin
until solution found
do DFS with depth cutoff c
c = c+1
End

Successive depth-first searches are conducted – each with depth bounds increasing by 1 as shown in the figure below:



Depth bound = 1 Depth bound = 2 Depth bound = 3 Depth bound = 4

Long Answer Type Questions

1. a) How are production system and control strategies applied in solving AI problems?

b) You are given two jars – *a* 4-litre one and *a* 3-litre one. Neither has any measuring mark on it. How can you get 2 litres of water into the 4-litre jug. With the help of state-space diagram, find a solution. **[WBUT 2006, 2007, 2012]**

OR,

You are given two jugs, a 4-gallon one and a 3-gallon one. Neither have any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 gallons of water into the 4-gallon jug? Give the state-space diagram, describe the production rules and give a possible solution. **[WBUT 2016]**

Answer:

a) Refer to Question No. 1 of Short Answer Type Questions.

b) State Space

The problem can be described as set of ordered pairs of integers (x, y)

x = 0, 1, 2, 3 or 4 where x number of gallons of water in the 4- gallon jug

y = 0, 1, 2, or 3 where y the quantity of water in the 3- gallon jug

start state : (0, 0)

goal state : (2, *n*) for any value of n

We can represent the problem using production rules. The general form of productions looks like this:

<antecendent1, antecedent2, ... antecedentN>
(condition1, condition2, ...)
---> (consequent1, consequent2, consequentM)
where the conditions on the antecedents specify the applicability of an operation

Production Rules

1 $(x,y) \rightarrow (4,y)$ if $x < 4$	Fill the 4-gallon jug
2 $(x,y) \rightarrow (x,3)$ if $y < 3$	Fill the 3-gallon jug
3 $(x,y) \rightarrow (x-d,y)$ if $x > 0$	Pour some water out of the 4-gallon jug
4 $(x,y) \rightarrow (x,y-d)$ if $x > 0$	Pour some water out of the 3-gallon jug
5 $(x,y) \rightarrow (0,y)$ if $x > 0$	Empty the 4-gallon jug on the ground
6 $(x,y) \rightarrow (x,0)$ if $y > 0$	Empty the 3-gallon jug on the ground
7 $(x,y) \rightarrow (4,y-(4-x))$ if $x + y \geq 4$ and $y > 0$	Pour water from the 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full
8 $(x,y) \rightarrow (x-(3-y),3)$ if $x + y \geq 3$ and $x > 0$	Pour water from the 4-gallon jug into the 3-gallon jug until the 3-gallon jug is full
9 $(x,y) \rightarrow (x+y, 0)$ if $x + y \leq 4$ and $y > 0$	Pour all the water from the 3-gallon jug into the 4-gallon jug
10 $(x,y) \rightarrow (0, x+y)$ if $x + y \leq 3$ and $x > 0$	Pour all the water from the 4-gallon jug into the 3-gallon jug

One solution

Gallons in the 4-Gallon Jug	Gallons in the 3-Gallon Jug	Rule Applied
0	0	2
0	3	9
3	0	2
3	3	7
4	2	5
0	2	9
2	0	

2. Discuss and compare hill climbing and best-first search technique.
[WBUT 2006, 2008, 2016]

OR,

Compare and contrast hill climbing and best-first search procedures. [WBUT 2009, 2012]

OR,

Compare and contrast Best-First and Hill climbing search. [WBUT 2018]

Answer:

1st Part:

Hill climbing technique & its Drawbacks:

Refer to Question No. 3 & 5 of Short Answer Type Questions.

2nd Part:

The **best first search** allows us to switch between paths thus gaining the benefit of both approaches. At each step the most promising node is chosen. If one of the nodes chosen generates nodes that are less promising it is possible to choose another at the same level and in effect the search changes from depth to breadth. If on analysis these are no better than this previously unexpanded node and branch is not forgotten and the search method reverts to the descendants of the first choice and proceeds, backtracking as it were.

This process is very similar to steepest ascent, but in hill climbing once a move is chosen and the others rejected the others is never reconsidered whilst in best first they are saved to enable revisits if an impasse occurs on the apparent best path. Also the best available state is selected in best first even its value is worse than the value of the node just explored whereas in hill climbing the progress stops if there are no better successor nodes. The best first search algorithm will involve an OR graph which avoids the problem of node duplication and assumes that each node has a parent link to give the best node from which it came and a link to all its successors. In this way if a better node is found this path can be propagated down to the successors. This method of using an OR graph requires 2 lists of nodes.

3. a) What do you mean by admissibility and consistency of a heuristic function? [WBUT 2007]

OR,

What do you mean by consistency of a heuristic? [WBUT 2016]

b) Validate each of the following statements giving brief explanation.

i) The heuristic function "Sum of Manhattan distances" for 8-puzzle problem is consistent.

ii) If heuristic is consistent then the heuristic is admissible but the converse is not true. [WBUT 2007]

Answer:

a) A heuristic h(n) is admissible if it never overestimates the cost to the goal from node n; i.e. it is always optimistic .

A heuristic h(n) is consistent if for any nodes A and B $h(B) >= h(A) + c(A,B)$

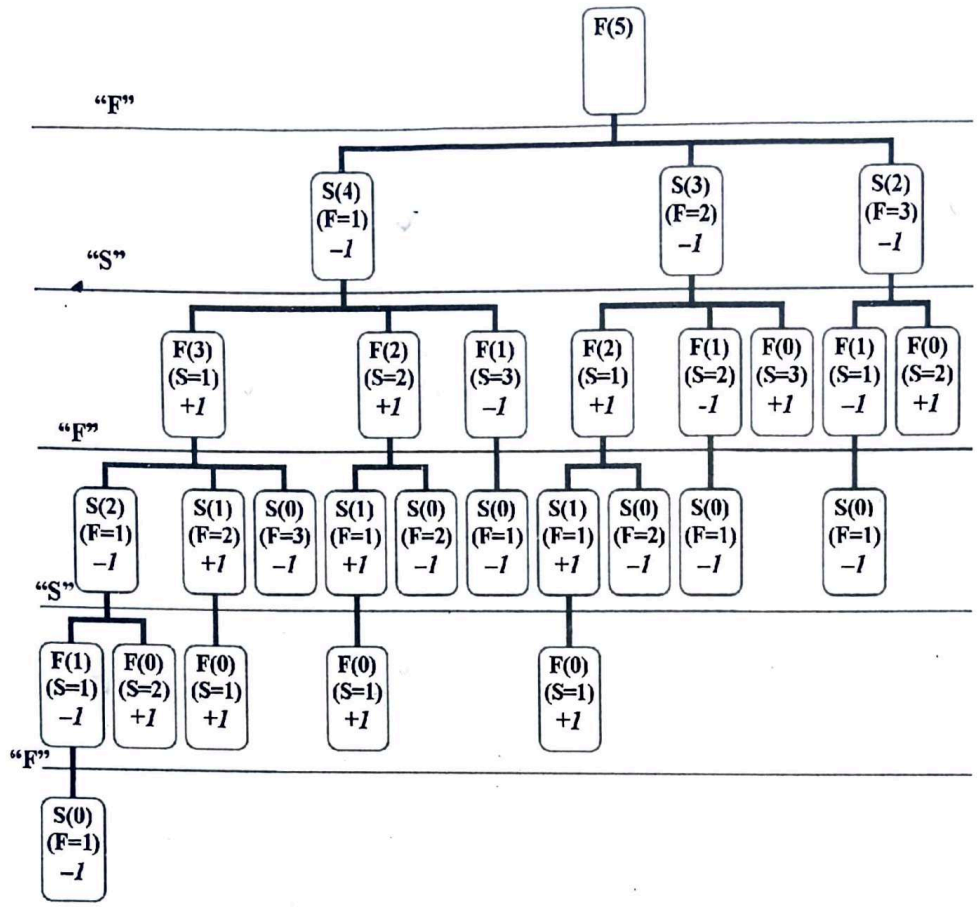
Intuitively, this says that our heuristic will become more accurate (less optimistic) as we approach the goal.

b) i) sum of Manhattan distances of tiles to their goal location is a heuristic that can be used for 8-puzzle problem. The Manhattan distance is the number of moves required if no other tiles are in the way. An heuristic is said to be consistent if one step takes us from n to n', then h(n) ≤ h(n') + cost of step from n to n'. The minimum number of moves to get from some position $<i_1,j_1>$ to some other position $<i_2,j_2>$ is the Manhattan distance, so this heuristic is consistent and A* is admissible.

ii) Proof: **If heuristic is consistent then the heuristic is admissible**
This is proved by induction on m, the length of the best path from node to goal. By assumption, $h(N_m) \le h^*(N_m)$, where $h^*(n)$ denotes the cost of the shortest path from n to the goal.
Therefore, $h(N_{m+1}) \le c(N_{m+1}, N_m) + h(N_m) \le c(N_{m+1}, N_m) + h^*(N_m) = h^*(N_{m+1})$ making it admissible. (N_{m+1} is any node whose best path to the goal, of length m + 1, goes through some immediate child N_m whose best path to the goal is of length m.)
Consider a search problem in which the states are nodes along a path $P = n_0, n_1 \ldots n_m$ where n_0 is the start state, n_m is the goal state and for all i there is an action from n_i to n_{i+1} of cost 1. The cheapest cost from any node n_i to the goal is then $k(n_i) = m - i$. Now we define a heuristic function $h(n) = m - 2 \times [i/2]$. Clearly this is admissible since for all states n_i, $h(n_i) \le k(n_i)$ but is not consistent.

4. The game of NIM is played as follows. Two players alternate in removing one, two, or three pennies from a stack initially containing five pennies. The player who picks up the last penny loses.
a) Draw the full game tree.
b) Show that the player who has the second move can always win.
c) Execute $\alpha - \beta$ pruning procedure on the game tree. How many terminal nodes are examined? **[WBUT 2007, 2012]**

Answer:
a) F(X) → X pieces left for first Player; if X=0, first player wins.
S(Y) → Y pieces left for second Player; if Y=0, second player wins.
F = a → F picks up a sticks.
S = b → S picks up b sticks.



b) Let us now assign +1 to each game that F wins and -1 to each game that S wins. The values are assigned to each leaf node. Using minmax method as shown in the figure, we conclude that S will win the game when both F and S use optimal strategies.
From the tree it is pretty clear that

i. when first player picks 3 sticks second player can pick 1 penny to ensure victory.
ii. when first player picks 2 sticks second player can pick 2 pennies to ensure victory.
iii. when first player picks 1 stick second player can pick 3 pennies to ensure victory.

c) Let us use alpha-beta pruning from left to right. Here we know that the value of each node can either be +1 or -1. It can be seen that 5 nodes are not expanded.

5. Consider the following 8-puzzle problem: **[WBUT 2009]**

Given the critical state:

2	8	3
1		4
7	6	5

and Goal the state:

1	2	3
8		4
7	6	5

i) List the operators.
ii) Select a heuristic function for the 8-puzzle problem
iii) Solve the problem by *A** algorithm with your selected heuristic function.

OR,

Consider the following problem:
The 8-puzzle consists if 3×3 board with 8 numbered tiles and a blank space. Each tile has a number on it. A tile adjacent to the blank space can only slide into that space. The game consists of a starting position and a specified goal position. The goal is to transform the starting position into the goal position by sliding the tiles around. Convert this problem to a state space search problem. Show that it works on the following example:

Start		
2	8	3
1	6	4
7		5

			Goal
2	2	3	
1		4	
7	6	5	

[WBUT 2014]

OR,

Consider the following arrangement and solve the problem using A* search. Define the State space, write the operations, define the heuristic and also find whether this heuristic is admissible or not. Also show the solution. [WBUT 2016, 2017]

Initial State:

2	8	3
1	6	4
7		5

Final State:

1	2	3
8		4
7	6	5

Answer:

i) *Operators*: blank moves left, right, up, or down.

ii) The "Tiles Out of Place heuristic" or "*Hamming priority function*" algorithm can be used.

The heuristic involves summing the number of tiles that are not in their correct location for the Goal State. This heuristic is admissible, because it is clear that any tile that is out of place must be moved at least once. Below is the result of this heuristic when tiles 1 to 8 are in the Start State as shown in the figure.

```
Tiles Out of Place = 1 + 1 + 0 + 0 + 0 + 0 + 0 + 1 = 3
```

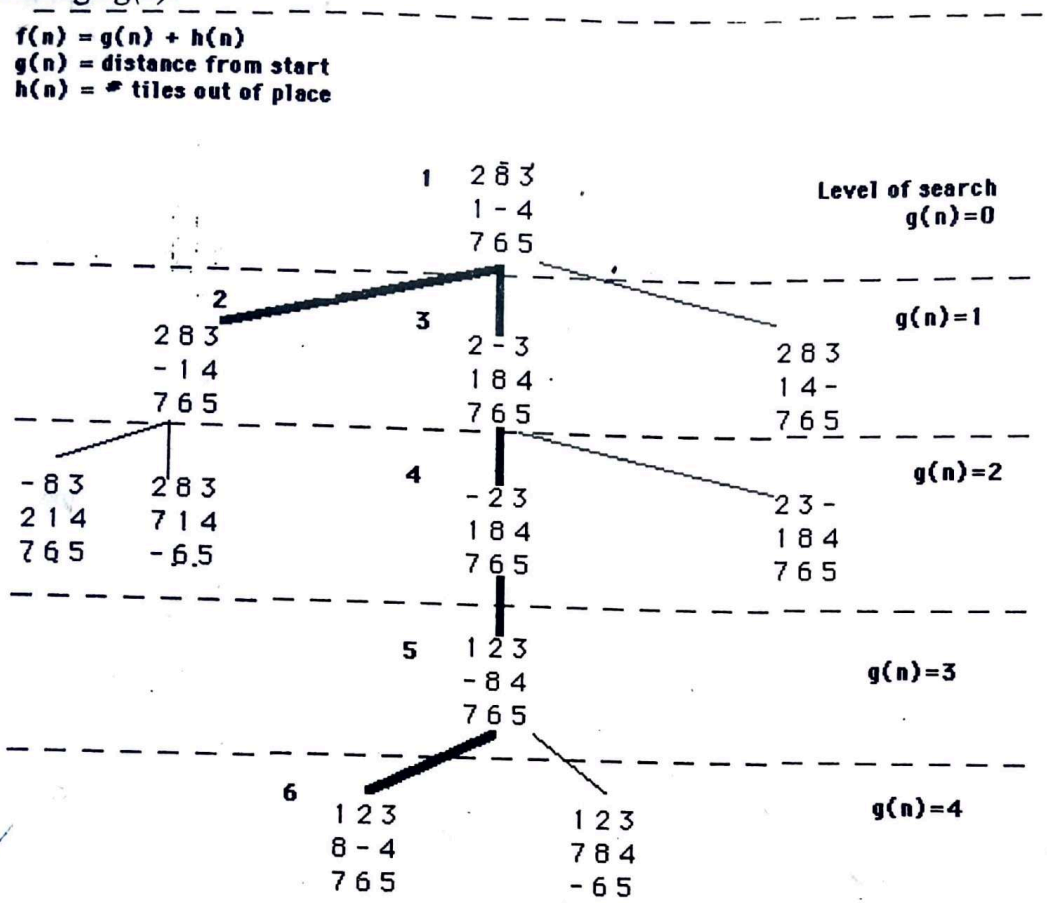
iii) A* uses the evaluation function:

f(n) = g(n) + h(n)

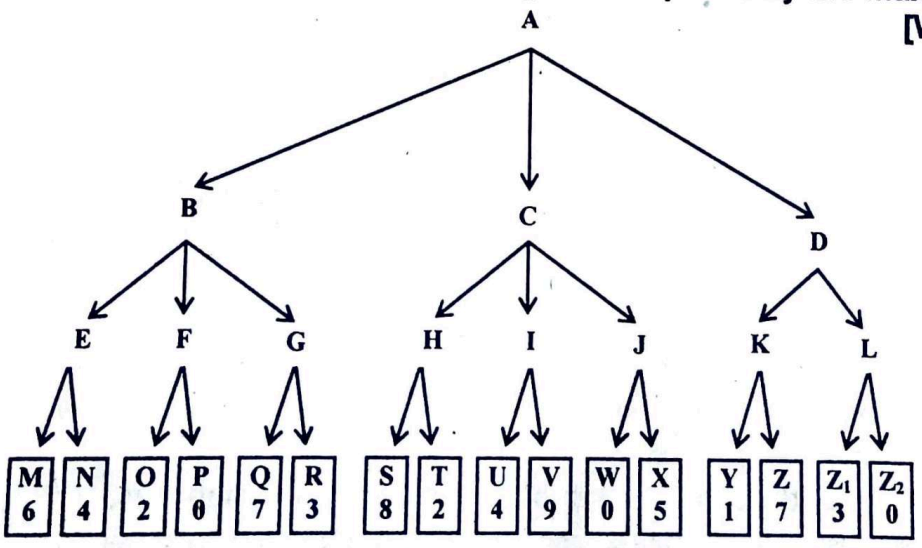
where *g(n)* measures the actual length of the path from the start state to the state *n*, and *h(n)* is a heuristic estimate of the distance from a state *n* to a goal state.

The following figure shows the full best-first search of the eight puzzle graph using the value of *h(n)* as the "number of tiles out of place". The number at the top of each state

represents the order in which it was taken off of the open list. The levels of the graph are used to assign $g(n)$.



6. Given a game tree for a two-ply game, where the evaluation function for winding are given at the leaf nodes. Assume that the game is opened by the maximizer. **[WBUT 2009]**



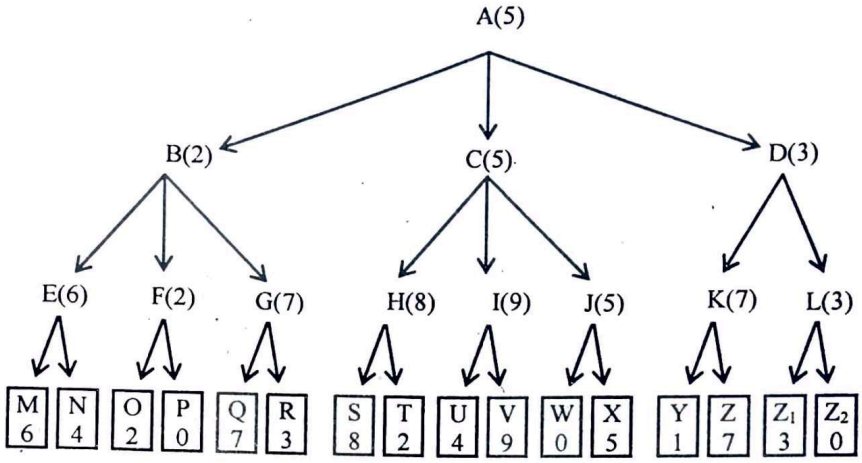
a) Using Minimax algorithm, determine which nodes the maximizer and the minimize should select in their first turn.

b) Identify the nodes that will be pruned by invoking Alpha-Beta algorithms.

OR,

Consider the following game tree where the evaluation function values for winning is given at leaf nodes. Assume that the game tree is opened by the maximizer.

[WBUT 2017]

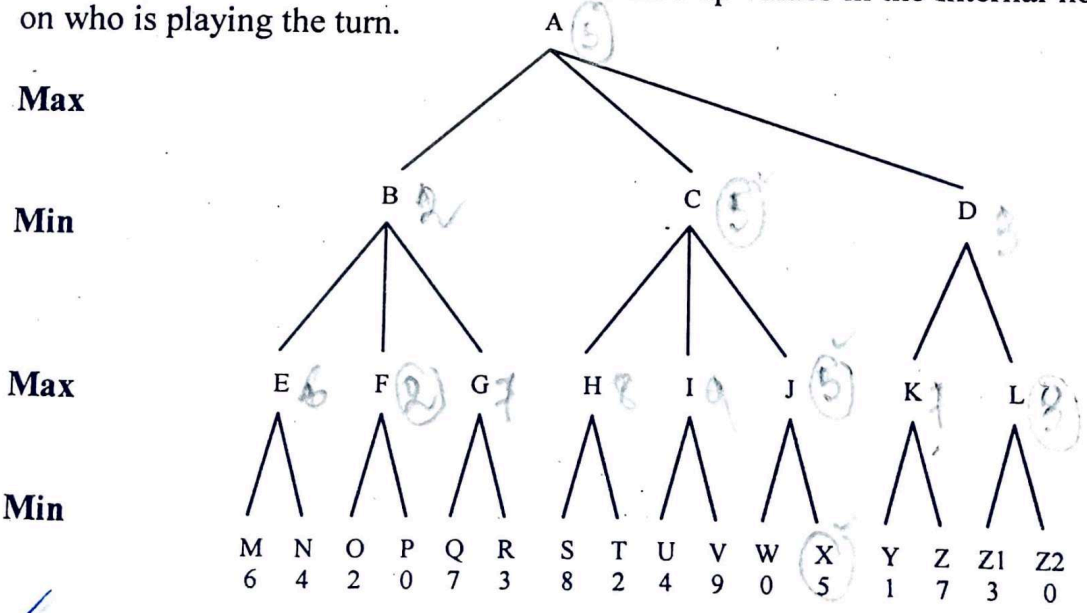


i) **Applying minimax algorithm determine which nodes the maximizer and minimizer would select in their turns.**

ii) **How many nodes will be pruned using alpha-beta pruning?**

Answer:

In the figure below we write down the backed up values in the internal nodes depending on who is playing the turn.



a) The moves are as follows:

Max chooses C

Min chooses J

Max chooses X

b) Node R will not be pruned and all the rest would be pruned.

7. Prove each of the following statements:

a) Breadth first search is a special case of uniform cost search. **[WBUT 2010, 2014, 2015]**

OR,

Is BFS identical to uniform cost search? Justify your answer. **[WBUT 2017]**

b) Breadth first, depth first and uniform cost search are special cases of Best First Search. **[WBUT 2010, 2014]**

c) Uniform cost search is a special case of A* search. **[WBUT 2010, 2015]**

Answer:

a) When all step costs are equal, the cost g(n) of a path from the start node to a node n is proportional to depth(n). Then uniform-cost search reproduces BFS with the modified search algorithm (a node is tested to be the goal when it is removed from the fringe).

b) Best-first search is the general term for an algorithm that expands the node with the least cost according to some evaluation function by keeping nodes to expand on a priority queue. Let f(n) is the cost function applied to node n.

Breadth first uses priority based on minimum path length and order of expansion. Thus Breadth first is best-first with f(n)=depth(n).

Depth first is prioritized on maximum path length and reverse order of expansion uniform-cost is prioritized on path cost from the start node. Thus Depth first is best-first with f(n)=-depth(n) or f(n) = 1/depth(n).

Uniform-cost is prioritized on path cost from the start node. Uniform-cost search expands the node with the lowest path cost.

Hence, breadth first, depth first and uniform cost search are special cases of Best First Search.

c) A* has the objective function = $g(n) + h(n)$, and uniform-cost is a special case with $h(n) = 0$ (thus f(n) = g(n); Contrariwise, A* with g(n) = 0 (thus f(n) = h(n)) is greedy.

8. a) How do you evaluate any search technique? **[WBUT 2011]**

Answer:

Evaluating the performance of an AI-based search technique can be complicated. We are concerned with mainly the measurements:

- How quickly a solution is found
- How good the solution is

There are several types of problems for which all that matters is that a solution, any solution, be found with the minimum effort. For these problems, the first measurement is especially important. In other situations, the quality of the solution is more important.

The speed of a search is affected both by the size of the search space and by the number of nodes actually traversed in the process of finding the solution. Because backtracking from dead ends is wasted effort, we want a search that seldom retraces its steps.

In AI-based searching, there is a difference between finding the best solution and finding a good solution. Finding the best solution can require an exhaustive search because sometimes this is the only way to know that the best solution has been found. Finding a

good solution, in contrast, means finding a solution that is within a set of constraints—it does not matter if a better solution might exist.
Some search techniques work better in certain situations than in others, so it is difficult to say whether one search method is always superior to another. But some search techniques have a greater probability of being better for the average case. In addition, the way a problem is defined can sometimes help one choose an appropriate search method.

b) Discuss on Bidirectional search technique. **[WBUT 2011]**

Answer:
Bidirectional Search, as the name implies, searches in two directions at the same time: one forward from the initial state and the other backward from the goal. This is usually done by expanding tree with branching factor b and the distance from start to goal is d. The search stops when searches from both directions meet in the middle. Bidirectional search is a brute-force search algorithm that requires an explicit goal state instead of simply a test for a goal condition. Once the search is over, the path from the initial state is then concatenated with the inverse of the path from the goal state to form the complete solution path.
Bidirectional search still guarantees optimal solutions. Assuring that the comparisons for identifying a common state between the two frontiers can be done in constant time per node by hashing. The time complexity of Bidirectional Search is O(b^d/2) since each search need only proceed to half the solution path. Since at least one of the searches must be breadth-first in order to find a common state, the space complexity of bidirectional search is also O(b^d/2). As a result, it is space bound in practice.

Advantages
The merit of bidirectional search is its speed. Sum of the time taken by two searches (forward and backward) is much less than the $O(b^d)$ complexity.
It requires less memory.

Disadvantages
Implementation of bidirectional search algorithm is difficult because additional logic must be included to decide which search tree to extend at each step.
One should have known the goal state in advance.
The algorithm must be too efficient to find the intersection of the two search trees.
It is not always possible to search backward through possible states.

9. a) Write down the disadvantages of hill climbing search procedure. [WBUT2011]

OR,

What are the three major problems of hill-climbing technique? [WBUT 2016]
b) When does simulated annealing algorithm behave like hill climbing? [WBUT 2011]

Answer:
The various disadvantages are:

1. **Local maxima**

A surface with two local maxima. (Only one of them is the global maximum.) If a hill-climber begins in a poor location, it may converge to the lower maximum.

A problem with hill climbing is that it will find only local maxima. Unless the heuristic is convex, it may not reach a global maximum. Other local search algorithms try to overcome this problem such as stochastic hill climbing, random walks and simulated annealing.

2. Ridges and alleys

Ridges are a challenging problem for hill climbers that optimize in continuous spaces. Because hill climbers only adjust one element in the vector at a time, each step will move in an axis-aligned direction. If the target function creates a narrow ridge that ascends in a non-axis-aligned direction (or if the goal is to minimize, a narrow alley that descends in a non-axis-aligned direction), then the hill climber can only ascend the ridge (or descend the alley) by zig-zagging. If the sides of the ridge (or alley) are very steep, then the hill climber may be forced to take very tiny steps as it zig-zags toward a better position. Thus, it may take an unreasonable length of time for it to ascend the ridge (or descend the alley).

By contrast, gradient descent methods can move in any direction that the ridge or alley may ascend or descend. Hence, gradient descent or conjugate gradient method is generally preferred over hill climbing when the target function is differentiable. Hill climbers, however, have the advantage of not requiring the target function to be differentiable, so hill climbers may be preferred when the target function is complex.

3. Plateau

Another problem that sometimes occurs with hill climbing is that of a plateau. A plateau is encountered when the search space is flat, or sufficiently flat that the value returned by the target function is indistinguishable from the value returned for nearby regions due to the precision used by the machine to represent its value. In such cases, the hill climber may not be able to determine in which direction it should step, and may wander in a direction that never leads to improvement.

b) If the starting temperature is not too hot then the ending solution will be very close to the starting solution. Thus this will become a simple climbing algorithm.

10. Consider the 3-puzzle problem shown in figure.

2	3
1	

Initial

1	2
3	

Final

Figure

Possible operators (in order) are: up, down, left, right; Assume that repeated states are not detected.

a) Draw the search tree using breadth first search. **[WBUT 2011, 2015]**

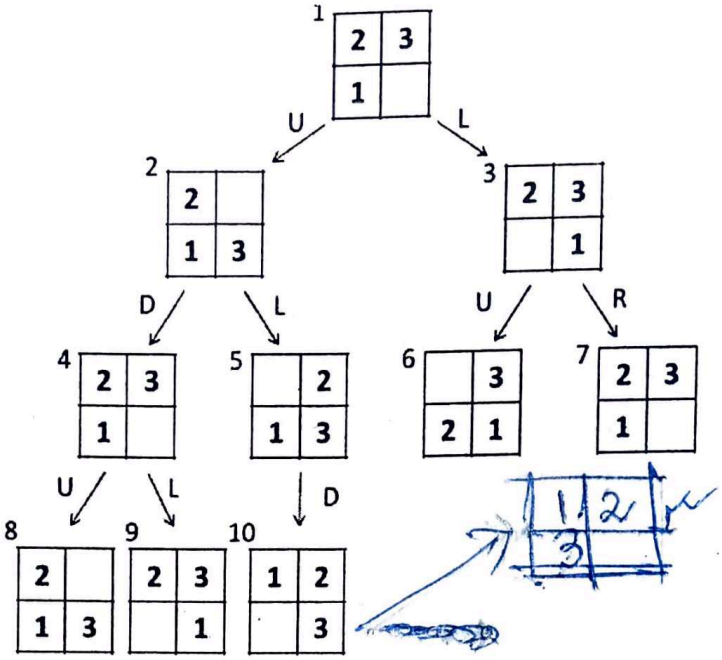
b) Would depth first search find the goal? Explain. **[WBUT 2011, 2015]**

c) How many nodes would be generated if Iterative Deepening is used starting with depth increment one? **[WBUT 2011]**

d) A* search with the heuristic being the number of misplaced tiles. **[WBUT 2015]**

Answer:

a) Breadth first search will start from the root node, then expands all the successors of the root node, and then all their successors and so on. Breadth first search stops when first solution is found.



b) Depth-first search always expands the deepest node in the search tree. Notice that there was no mechanism to remember states that have been visited earlier. Depth first will not find a solution as it will start oscillating between movements U and D.

c) Iterative deepening will prevent the looping.

The no. of nodes generated would be: $(d+1)b^0 + (d)b^1 + (d-1)b^2 + \ldots b^d$

where b is the branching factor and d is the depth.

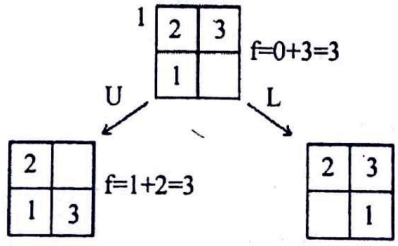
d) $f(n) = g(n) + h(n)$

$h(n)$ = no. Of misplaced tiles

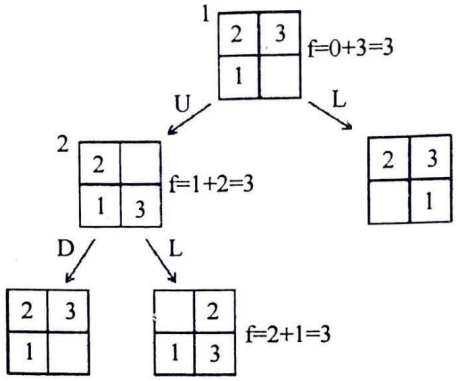
$g(n)$= distance from start

At the start state, h=-3 (all 3 tiles are misplaced)

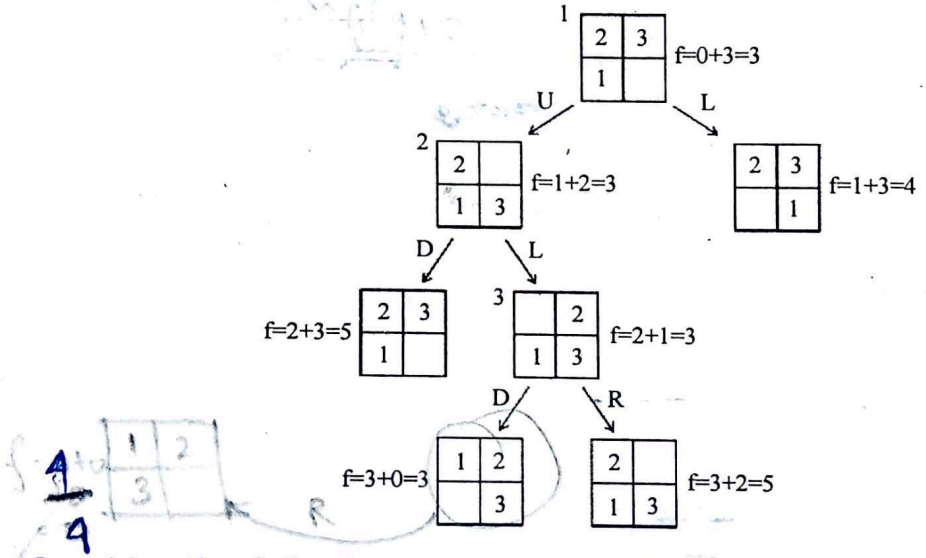
When the start state is opened we see two states as shown below.



The leftmost state has a lower cost so that one is chosen and opened. We now see states with costs 5, 3 and 4 as shown below.



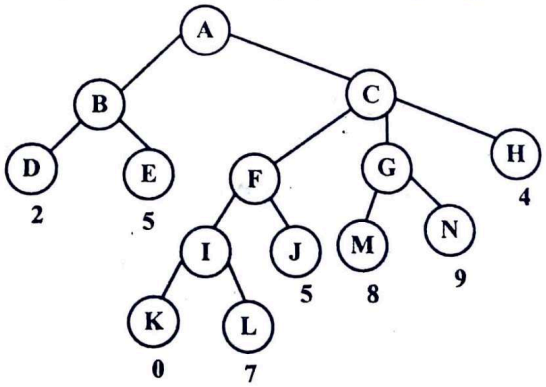
The state with lowest cost is opened and we see two more nodes including the goal node as shown below. We notice that the goal node has the lowest cost so we choose that and can finish the search. If some other node with a lower cost function value was still visible, A* search would choose that instead of the higher cost goal node. This is because A* tries to find the path with lowest cost.



11. Consider the following game tree in which static scores are all from first player's point of view. **[WBUT 2011]**

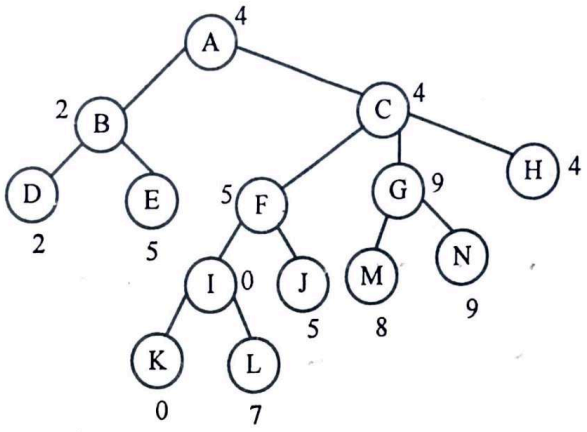
i) Which would be his best first move if MINIMAX algorithm is used?

ii) Which branches will be pruned if $\alpha - \beta$ pruning algorithm is used?

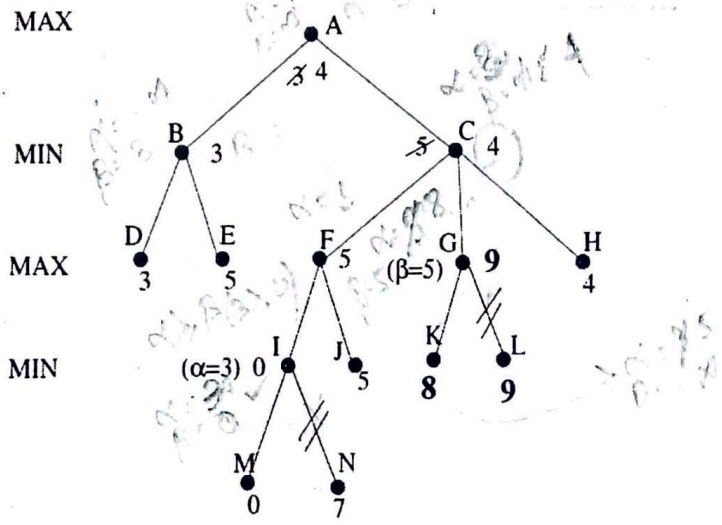


Answer:

i) The following graph shows the values associated with each of the nodes using MINIMAX algorithm. The best move for the first layer will be C.



ii) Alpha-beta pruning left to right



12. a) Consider the following problem:

A farmer is on the left bank of a river with a boat, a cabbage, a goat, and a wolf. The task is to get everything to the right bank of the river. The restrictions are as follows:

Only the farmer can handle the boat, when he is in the boat, there is only space for one more item and the farmer can't leave the goat alone either with the wolf, or with the cabbage.

Represent this problem as a state space search problem & show at least one solution path.

b) Define admissible & consistent heuristic. For a heuristic h, prove the following:

i) if h is consistent, then prove that $h(n) \le c(n,n') + h(n')$ is applicable for any descendent n' of n.

ii) if h is consistent, then prove that h is admissible also. **[WBUT 2013]**

Answer:

a) In this problem information can be represented in the form of the following 4-tuple. : (A, B, C, D) for farmer, wolf, goat and cabbage respectively. Each variable position stores the location of a particular object. For instance, the initial state where the farmer, the goat, the wolf and the cabbage are on the south bank is given by (S,S,S,S). The goal state is represented by (N,N,N,N).

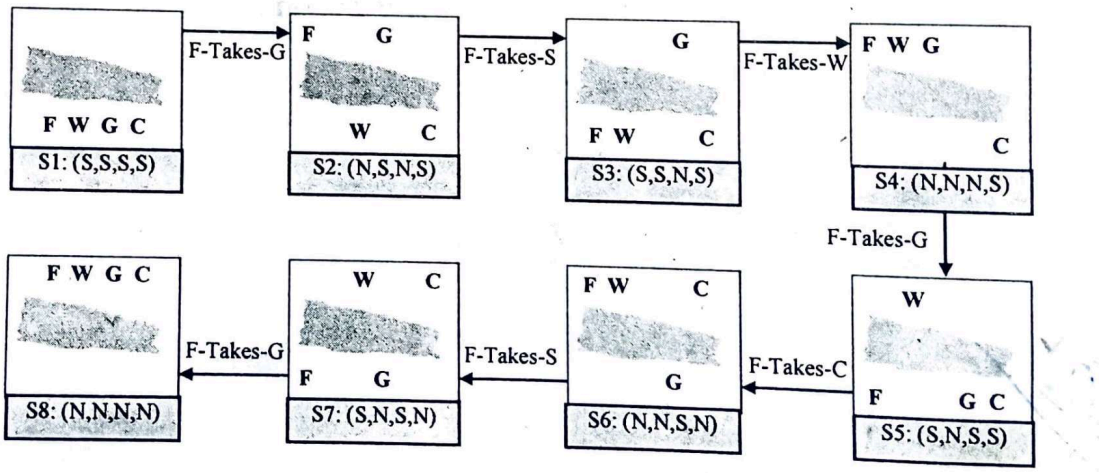
- There are 4 transforming operators:
  - Farmer-Takes-Self

    (A,B,C,D) (Opposite(A), B,C,D?)
  - Farmer-Takes-Wolf

    (A,B,C,D) (Opposite(A), Opposite(B),C,D)
  - Farmer-Takes-Goat

    (A,B,C,D) (Opposite(A), B, Opposite(C),D)
  - Farmer-Takes-Cabbage

    (A,B,C,D) (Opposite(A), B,C, Opposite(D))
- Finally, a state is safe as long as the wolf and the goat or the cabbage are not left unattended – this is the precondition to the above operators. While there are other possibilities here is one 7 step solution to the river problem.



b) A heuristic function is said to be **admissible** if it never overestimates the cost of reaching the goal, i.e. the cost it estimates to reach the goal is not higher than the lowest possible cost from the current point in the path. Formally, a heuristic *h(n)* is admissible if for every node *n*, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from *n*.
A consistent (or monotone) heuristic function is a function that estimates the distance of a given state to a goal state, and that is always at most equal to the estimated distance from any neighboring vertex plus the step cost of reaching that neighbor.
Formally, for every node N and every successor P of N generated by any action a, the estimated cost of reaching the goal from N is no greater than the step cost of getting to P plus the estimated cost of reaching the goal from P. In other words:

$$h(N) \leq c(N,P) + h(P)$$

where
h is the consistent heuristic function
N is any node in the graph
P is any descendant of N
G is any goal node
c(N,P) is the cost of reaching node P from N

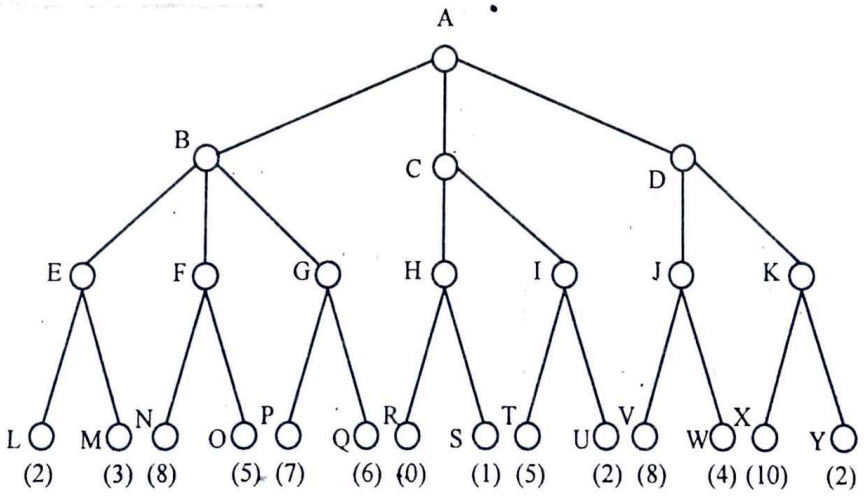
i) Let k(n) be the cost of the cheapest path from n to the goal node. We will prove by induction on the number of steps to the goal that h(n) <= k(n).
Base case: If there are 0 steps to the goal from node n, then n is a goal and therefore h(n) = 0 <= k(n).

Induction step: If n is i steps away from the goal, there must exist some successor n' of n generated by some action a such that n' is on the optimal path from n to the goal (via action a) and n' is i − 1 steps away from the goal.

Therefore, h(n) <=c(n, n') + h(n')
But by the induction hypothesis, h(n') <= k(n').
Therefore, h(n) <= c(n, n') + k(n') = k(n)
since n' is on the optimal path from n to the goal via action a.

ii) ***Refer to Long Answer Type Question No. 3(b)(ii).***

13. a) Define Constraint Satisfaction Problem. 4-queens problem seeks to place 4-queens in a 4×4 chess board such that no two queens will attack each other in either horizontal or vertical or diagonal way. Formulate this problem as CSP.
b) Consider the following game tree in which the static scores (in parentheses at the leaf nodes) are all from the first player's point of view. Assume that the first player is the maximizing player.



i) **What move should the first player choose?**
ii) **What nodes would not need to be examined using alpha-beta cutoff algorithm, assuming that nodes are examined in left-to-right order?** **[WBUT 2013]**

Answer:

a) 1st Part:

Constraint satisfaction problems (or CSPs) consist of variables with constraints on them. Many important real-world problems can be described as CSPs.

Examples of some of the most common problems include: The n-Queen problem i.e., the local condition is that no two queens attack each other, i.e. are on the same row, or column, or diagonal, a cryptography problem, map colouring problem etc. Formally speaking, a constraint satisfaction problem (or CSP) is defined by a set of variables, X1;X2; : : : ;Xn, and a set of constraints, C1;C2; : : : ;Cm. Each variable Xi has a nonempty domain Di of possible values. Each constraint Ci involves some subset of the variables and specifies the allowable combinations of values for that subset. A state of the problem is defined by an assignment of values to some or all of the variables, {Xi = vi, Xj =vj ,...} An assignment that does not violate any constraints is called a consistent or legal assignment. A complete assignment is one in which every variable is mentioned, and a solution to a CSP is a complete assignment that satisfies all the constraints. Some CSPs also require a solution that maximizes an objective function.

2nd Part:

4 queens problem as CSP

Variables: Qi (i is the column)

- Domains: Di = {1, 2, 3, 4} (the row)
- Constraints

– Qi≠Qj (cannot be in same row)

– |Qi - Qj| ≠ |i - j| (or same diagonal)

b) (i) Evaluating the minimax algorithm, we find the values at all the node by moving upwards from the leaf node.

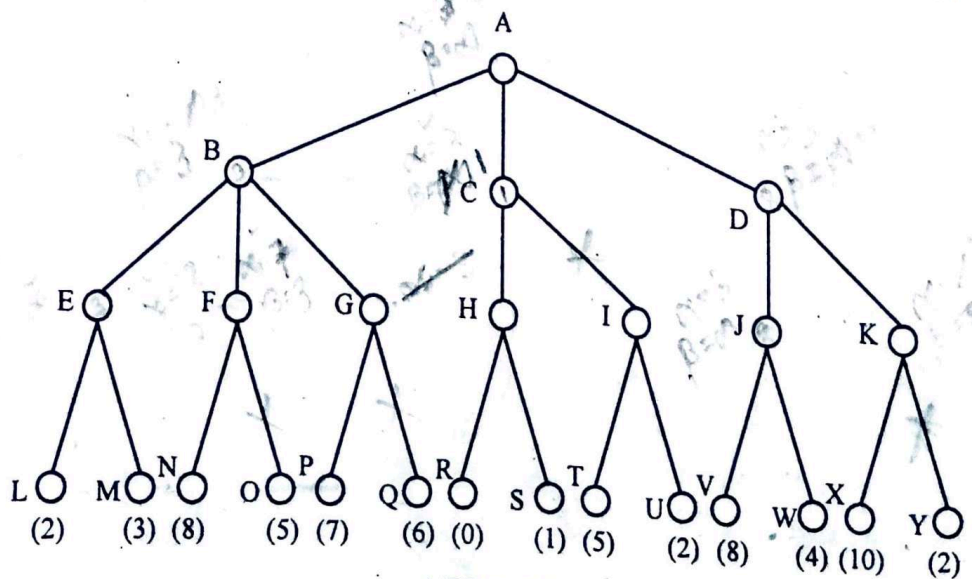
E=3, F=8, G=7, H=1, I=5, J=8, K=10

B=3, C=1, D= 8

A=8

We therefore find that the 1st player will choose D, which gives an outcome of 8 at node V.

(ii)



While examining the nodes from left to right MAX will explore L, M and N. Since value at N=8 and E has a value 3, it will not examine O. Again since P=7 which is more than E, MAX will not examine Q. So value at B =3. Again nodes R and S are examined and since the value at H=1. Thus MAX will ignore I, T and U since B=3 and H=1. Again similarly after examining V, W and X the node left out will be Y.
Hence the nodes that are not examined are: O, Q, I, T, U, Y

14. A farmer has a wolf, goat and cabbage. He wants to cross a river with all its possessions. He has a single boat that can carry at most one of his possessions on any trip. Moreover, an unattended wolf eats goat and unattended goat eats cabbage. How does he do the transportation?
Describe a production system stating clearly the production rules and control strategies. Now, show that your proposed production system solution can solve this problem. **[WBUT 2014]**

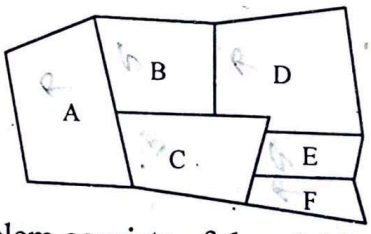
Answer:
States of the problem space are represented using the predicate
state(F,W,G,C)
Initial state given by the fact state(w,w,w,w) and goal state is (e,e,e,e) where w and e represent west and east of the river.
Rule to operate on state with F and W on the same side to produce state with
F and W on the other side :
move(state(X,X,G,C), state(Y,Y,G,C)) : -opp(X,Y).
opp(e,w).
/* ensures e/w swap */
opp(w,e).
/* ensures w/e swap *
The unsafe states may be represented by the rules
/* wolf eats goat */
unsafe(state(X,Y,Y,C)) :- opp(X,Y).
/* goat eats cabbage */
unsafe(state(X,W,Y,Y)) : - opp(X,Y)
Using above rules to modify the move rule:
/* farmer takes wolf to other side */
(1)
move(state(X,X,G,C), state(Y,Y,G,C)):
opp(X,Y), not (unsafe(state(Y,Y,G,C))).
Similarly,
/* farmer takes goat to other side */
(2)
move(state(X,W,X,C), state(Y,W,Y,C)):
opp(X,Y), not (unsafe(state(Y,W,Y,C))).
/* farmer takes cabbage to other side */
(3)
move(state(X,W,G,X), state(Y,W,G,Y)):

```
opp(X,Y), not (unsafe(state(Y,W,G,Y))).
/* farmer takes himself to other side */
(4)
move(state(X,W,G,C), state(Y,W,G,C)):
opp(X,Y), not (unsafe(state(Y,W,G,C))).
/* gets here when none of the above fires, system predicate 'fail' causes a backtrack*/
move(state(F,W,G,C),  state(F,W,G,C)): fail.
```

Following the above rules:

Try farmer takes goat	e w e w
Try farmer takes self	w w e w
Try farmer takes wolf	e e e w
Try farmer takes goat	w e w w
Try farmer takes cabbage	e e w e
Try farmer takes wolf	w w w e
Try farmer takes goat	e w e e
BACKTRACK from	e w e e
BACKTRACK from	w w w e
Try farmer takes self	w e w e
Try farmer takes goat	e e e e

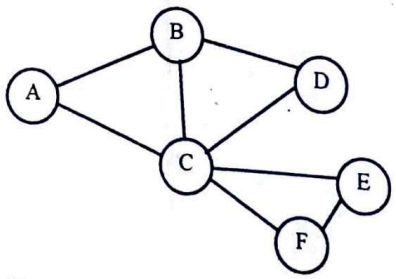
15. Consider the "map colouring problem" where a given map is to be coloured in a manner so that no neighbouring states of a country contain the same colour. Give a solution to following map colouring problem viewing it as a Constraint Satisfaction Problem. **[WBUT 2014]**



Answer:

The constraint satisfaction problem consists of 6 variables {A,B,C,D,E,F}. Each variable has the same domain {red, green, blue}. No two adjacent variables have the same value, so the constraints are: A≠B, A≠C, B≠C, B≠D, C≠D, C≠E, C≠F,D≠E, E≠F. Let us consider the domain as {Red, Green, Blue}.

This can be represented using a constraint graph as follows:



There are only 18 solutions (valid assignments). They can be counted as follows: Fix the color of SA (chosen because it is the most constraining variable). For each color of SA it remains 2 valid colors for each state, except T that still has 3. So let the backtracking algorithm a value for WA. Each color of SA (combined with the color already chosen for

WA) leaves only one color for NT, then one color for Q, then one for NSW, then one for V. So, there are 6 valid assignments for the states.
Final Solution could be:
A: Red, B: Green, C: Blue, D: Red, E: Green, F: Red

16. Prove that A* is admissible. **[WBUT 2015]**

Answer:

A* is both admissible and considers fewer nodes than any other admissible search algorithm with the same heuristic, because A* works from an "optimistic" estimate of the cost of a path through every node that it considers — optimistic in that the true cost of a path through that node to the goal will be at least as great as the estimate. But, critically, as far as A* "knows", that optimistic estimate might be achievable.
When A* terminates its search, it has, by definition, found a path whose actual cost is lower than the estimated cost of any path through any open node. But since those estimates are optimistic, A* can safely ignore those nodes. In other words, A* will never overlook the possibility of a lower-cost path and so is admissible.
Suppose now that some other search algorithm B terminates its search with a path whose actual cost is not less than the estimated cost of a path through some open node. Algorithm B cannot rule out the possibility, based on the heuristic information it has, that a path through that node might have a lower cost. So while B might consider fewer nodes than A*, it cannot be admissible. Accordingly, A* considers the fewest nodes of any admissible search algorithm that uses a no more accurate heuristic estimate.

17. Consider the following game tree. **[WBUT 2015]**

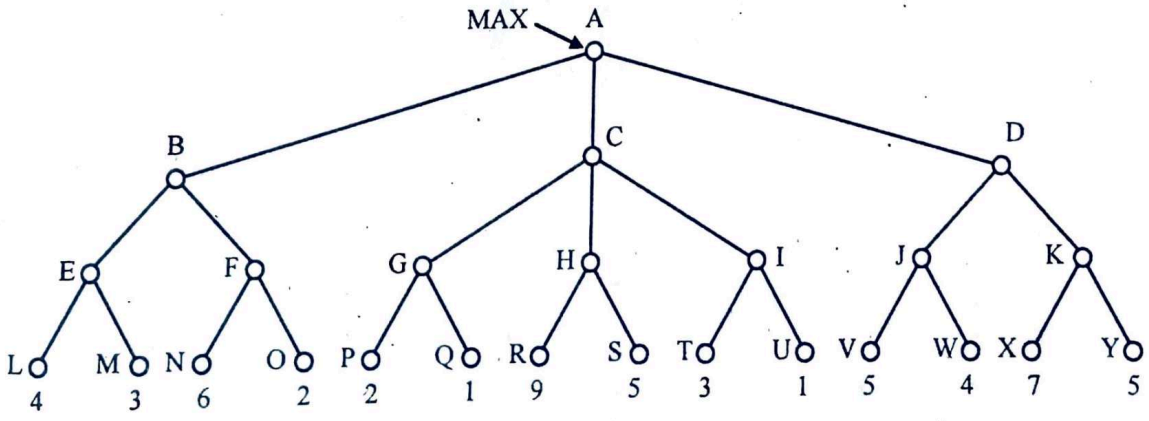
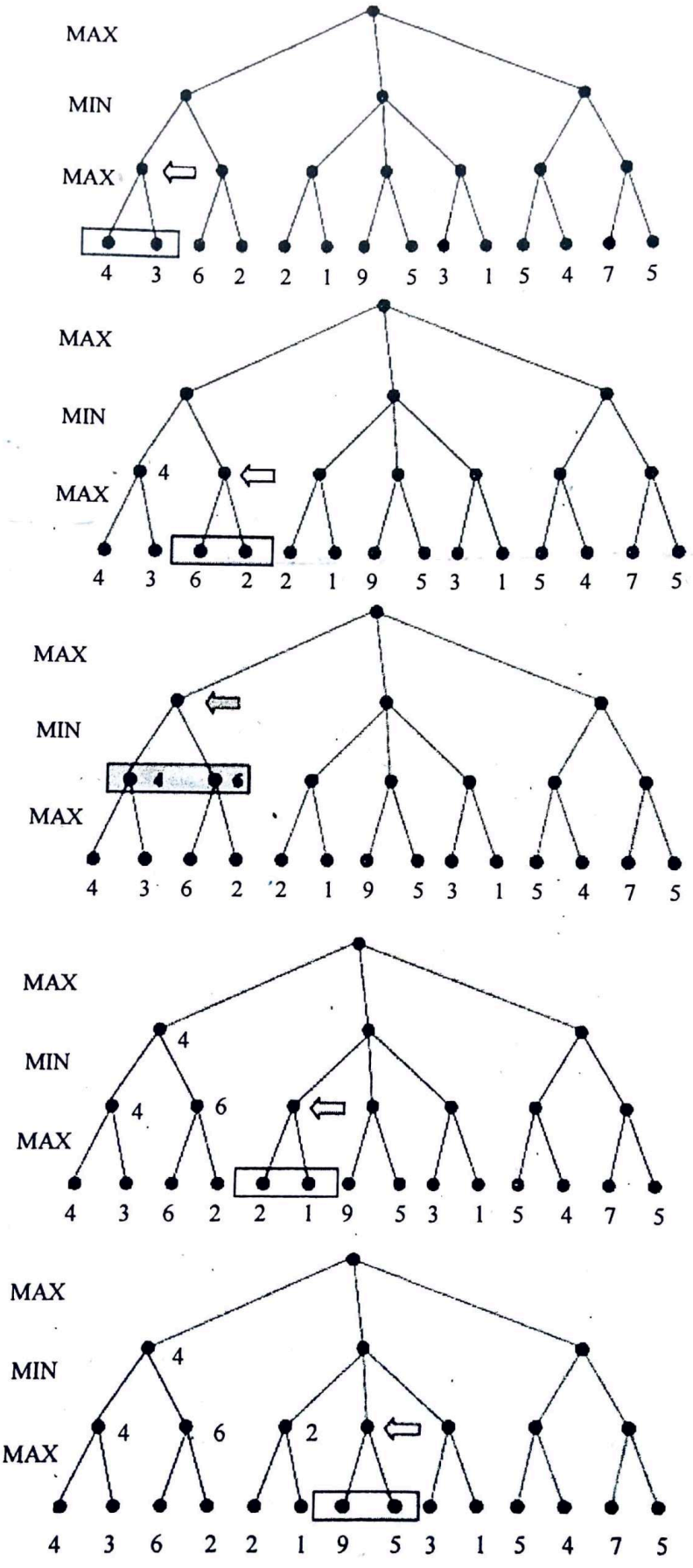


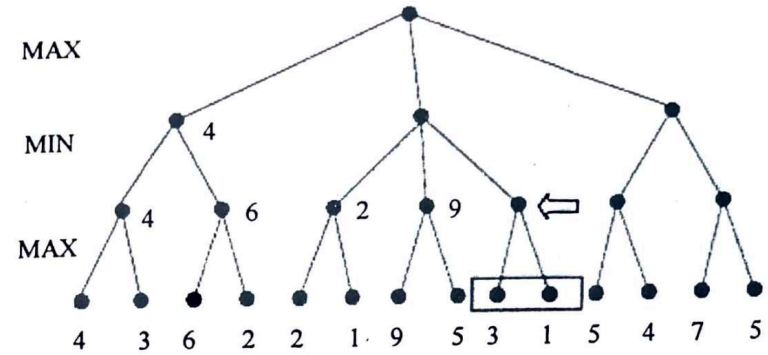
Fig: 1

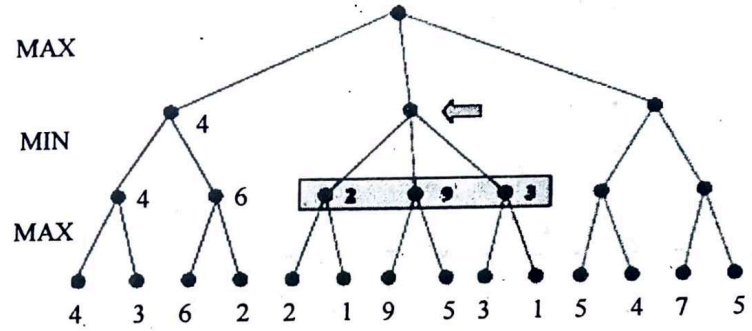
(i) Using MINIMAX procedure, determine what moves should be chosen by the maximizer in his first turn.
(ii) Execute Alpha-Beta pruning on the above game tree. How many terminal nodes are examined? For each cutoff specify whether it is an Alpha-cutoff or Beta-cutoff.

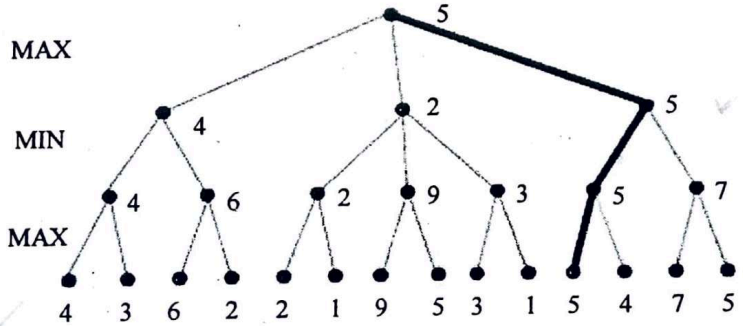
Answer:

i) The steps for minimax is as shown below.


MAX
MIN
MAX
4 3 6 2 2 1 9 5 3 1 5 4 7 5
MAX
MIN
4
MAX
4 3 6 2 2 1 9 5 3 1 5 4 7 5
MAX
MIN
MAX
4 3 6 2 2 1 9 5 3 1 5 4 7 5
MAX
4
MIN
4 6
MAX
4 3 6 2 2 1 9 5 3 1 5 4 7 5
MAX
4
MIN
4 6 2
MAX
4 3 6 2 2 1 9 5 3 1 5 4 7 5

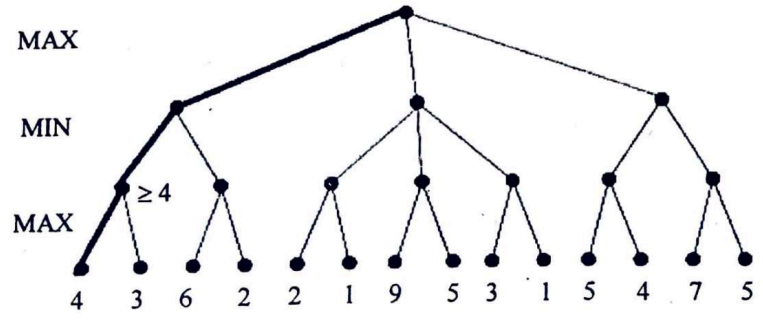


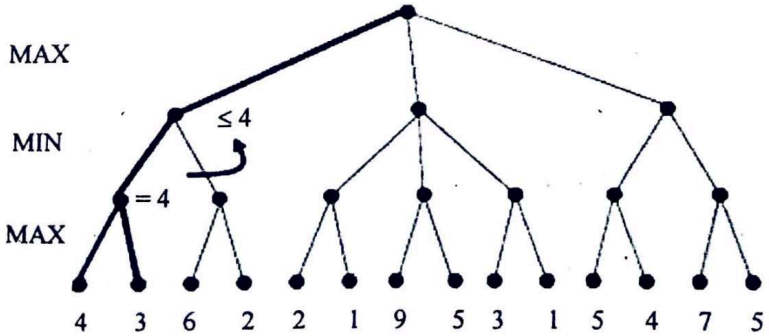


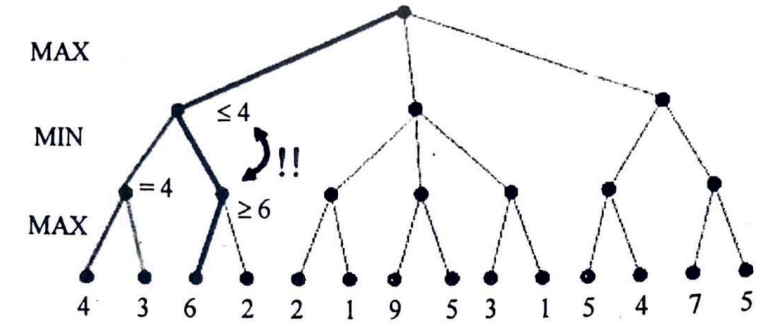


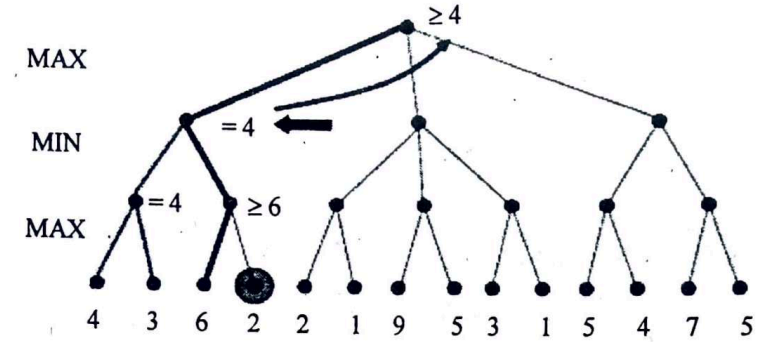
Thus the maximizer chooses node D.

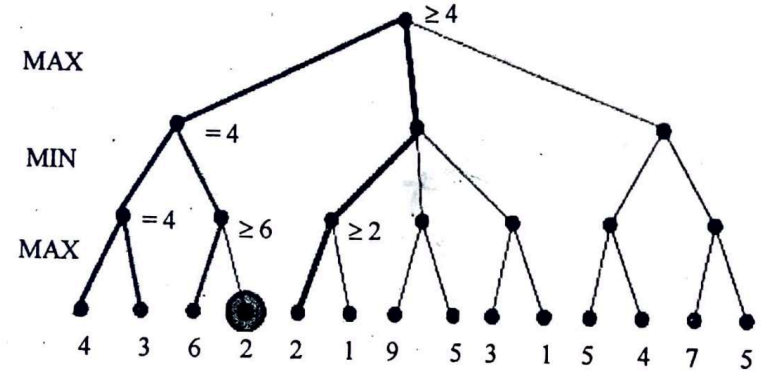
ii) The steps for alpha-beta pruning are shown below:

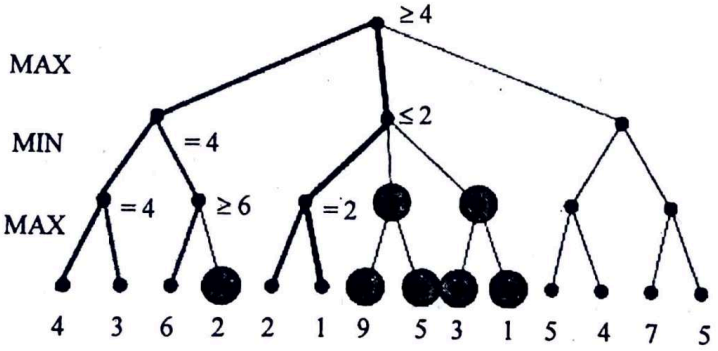


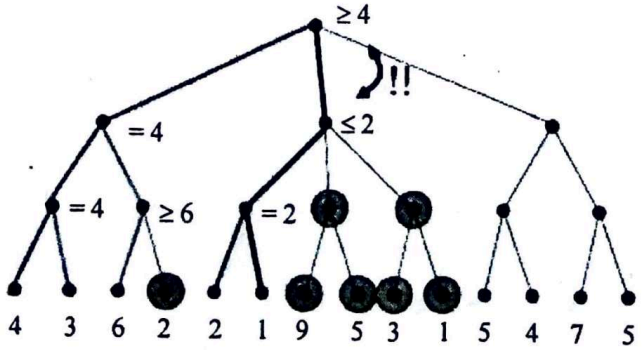


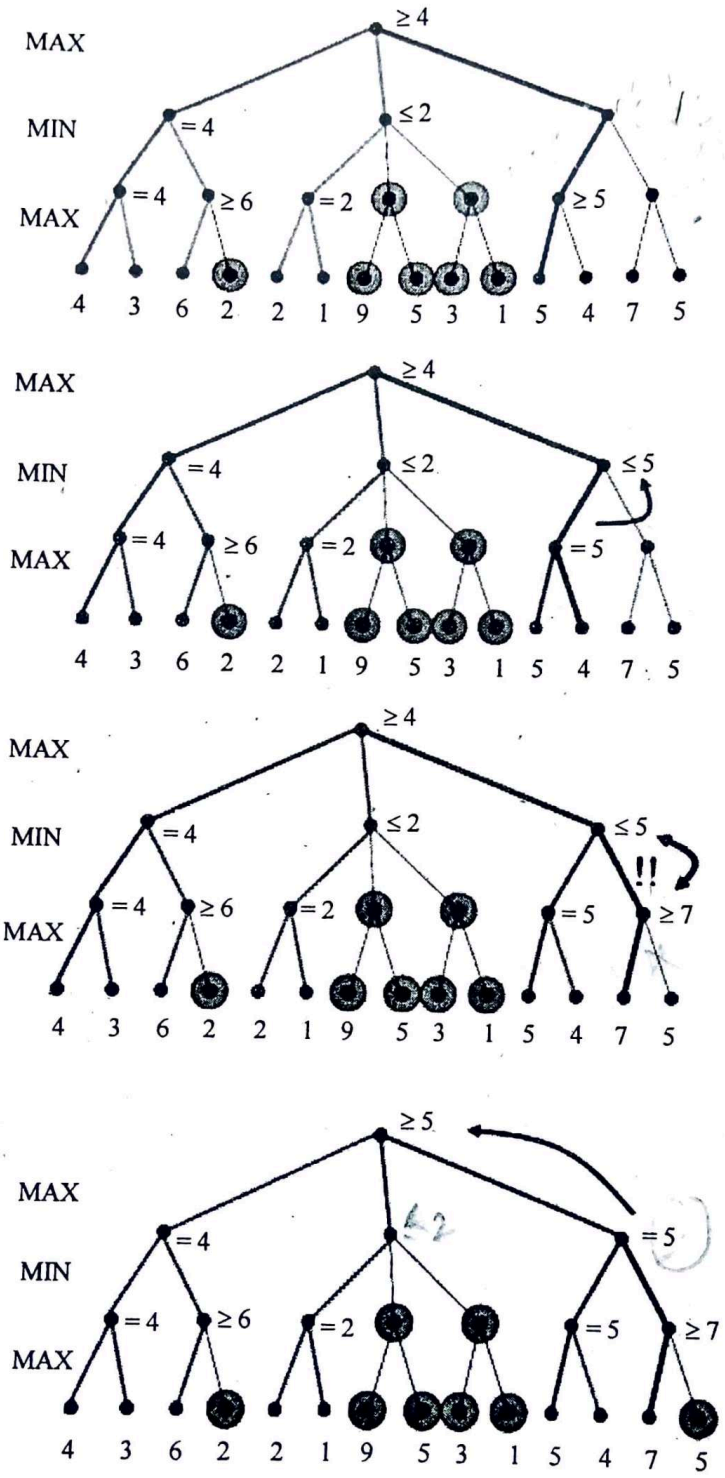

MAX
MIN
MAX
≤ 4
!!
= 4
≥ 6
4 3 6 2 2 1 9 5 3 1 5 4 7 5

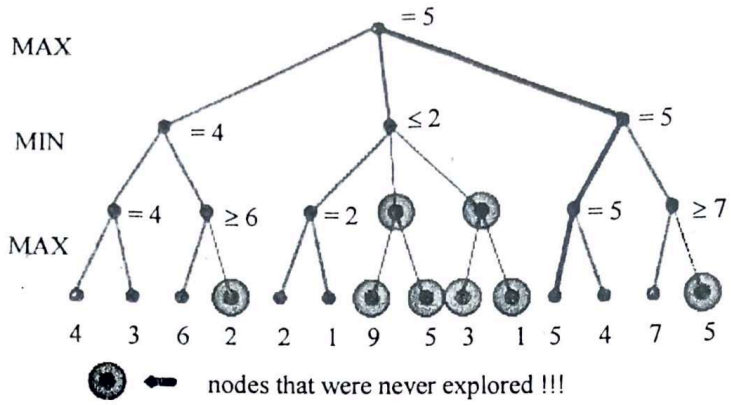

≥ 4
MAX
MIN
MAX
= 4
= 4
≥ 6
4 3 6 2 2 1 9 5 3 1 5 4 7 5


≥ 4
MAX
MIN
MAX
= 4
= 4
≥ 6
≥ 2
4 3 6 2 2 1 9 5 3 1 5 4 7 5


≥ 4
MAX
MIN
MAX
≤ 2
= 4
= 4
≥ 6
= 2
4 3 6 2 2 1 9 5 3 1 5 4 7 5


≥ 4
!!
≤ 2
= 4
= 4
≥ 6
= 2
4 3 6 2 2 1 9 5 3 1 5 4 7 5


MAX
MIN
MAX
≥ 4
= 4
≤ 2
= 4
≥ 6
= 2
≥ 5
4 3 6 2 2 1 9 5 3 1 5 4 7 5
MAX
MIN
MAX
≥ 4
= 4
≤ 2
≤ 5
= 4
≥ 6
= 2
= 5
4 3 6 2 2 1 9 5 3 1 5 4 7 5
MAX
MIN
MAX
≥ 4
= 4
≤ 2
≤ 5
!!
= 4
≥ 6
= 2
= 5
≥ 7
4 3 6 2 2 1 9 5 3 1 5 4 7 5
MAX
MIN
MAX
≥ 5
= 4
= 5
= 4
≥ 6
= 2
= 5
≥ 7
4 3 6 2 2 1 9 5 3 1 5 4 7 5



Alpha-cutoff occurs at when Min makes a move and Beta-cutoff occurs when Max makes a move as shown in the figure above.

18. a) Is uniform cost search a special case of Best First Search? Justify your answer.
b) Describe local beam search.
c) Three missionaries and three cannibals are standing at the left bank of a river. There is a boat having a capacity of taking two people and it can be driven by a missionary or a cannibal. If the number of missionaries is less than the number of cannibals at any bank, then cannibal will eat missionary. How is it possible for all the missionaries and cannibals to cross the river so that no missionary is getting eaten? Describe the state space of the problem. Describe the production rules for solving the problem. Show one solution of the problem. [WBUT 2017]

Answer:

a) ***Refer to Question No. 7 (b) of Long Answer Type Questions.***

b) In "local beam search", the search begins with k randomly generated states and does the following:
- At each step, all the successors of all k states are generated
- If any one of the successors is a goal, the algorithm halts
- Otherwise, it selects the k best successors from the complete list and repeats
- The parallel search of beam search leads quickly to abandoning unfruitful searches and moves its resources to where the most progress is being made

c) **Each state space can be represented by**
State(no_of_missionaries, no_of_cannibals, side_of_the_boat)
where no_of_missonaries are the number of missionaries at left side of river, no_of_cannibals are the number of cannibals at the left side of river and side_of_the_boat is the side of the boat at particular state.
For our case
Initial State => State(3, 3, L) and
Final State => State(0, 0, R).
Where L represents left side and R represents right side of river.

Production Rules for Missionaries and Cannibals problem.

- Rule 1: (i, j) : Two missionaries can go only when i-2>=j or i-2=0 in one bank and i+2>=j in the other bank.
- Rule 2: (i, j) : Two cannibals can cross the river only when j-2<=i or i=0 in one bank and j+2 <= i or i+0 or i=0 in the other.
- Rule 3: (i, j) : One missionary and one cannibal can go in a boat only when i-1>=j-1 or i=0 in one bank and i+1>=j+1 or i=0 in the other.
- Rule 4: (i, j) : one missionary can cross the river only when i-1>=j or i=0 in one bank and i+1>=j in the other bank.
- Rule 5: (i, j) : One cannibal can cross the river only when j-1 <=i or i=0 in one bank and j+1<=i or j=0 in the other bank of the river.

Let missionaries, cannibals and boat be denoted by M, C and B. The following table gives one possible solution.

	Left side	*Right side*
0. Initial setup:	CCCMMM B	
1. Two cannibals *cross over:*	MMMC	B CC
2. One comes back:	MMMCC	B C
3. Two cannibals go over again:	MMM	B CCC
4. One cannibal comes back:	MMMC	B CC
5. Two missionaries *cross:*	MC	B MMCC
6. A missionary *& a* cannibal *return:*	MMCC	B MC
7. Two missionaries *cross again:*	CC	B MMMC
8. A cannibal *returns:*	CCC	B MMM
9. Two cannibals *cross:*	C	B MMMCC
10. One cannibal returns:	CC	B MMMC
11. And brings over the third cannibal:	—	B MMMCCC

19. What do you mean by constraint satisfaction problem? Solve the following cryptography problem using constraint satisfaction search: [WBUT 2018]

S E N D
M O R E
=========
M O N E Y

Answer:

1st part: ***Refer to Question No. 13(a) (1st Part) of Long Answer Type Questions.***

2nd part:

Constraints:

- No two digits can be assigned to same letter.
- Only single digit number can be assigned to a letter.
- No two letters can be assigned same digit.
- Rule of arithmetic may be followed.

Initial state of problem.

D=?
E=?
Y=?
N=?
R=?
O=?
S=?
M=?
C1=?
C2=?

C1, C2, C3 stands for the carry variables respectively.

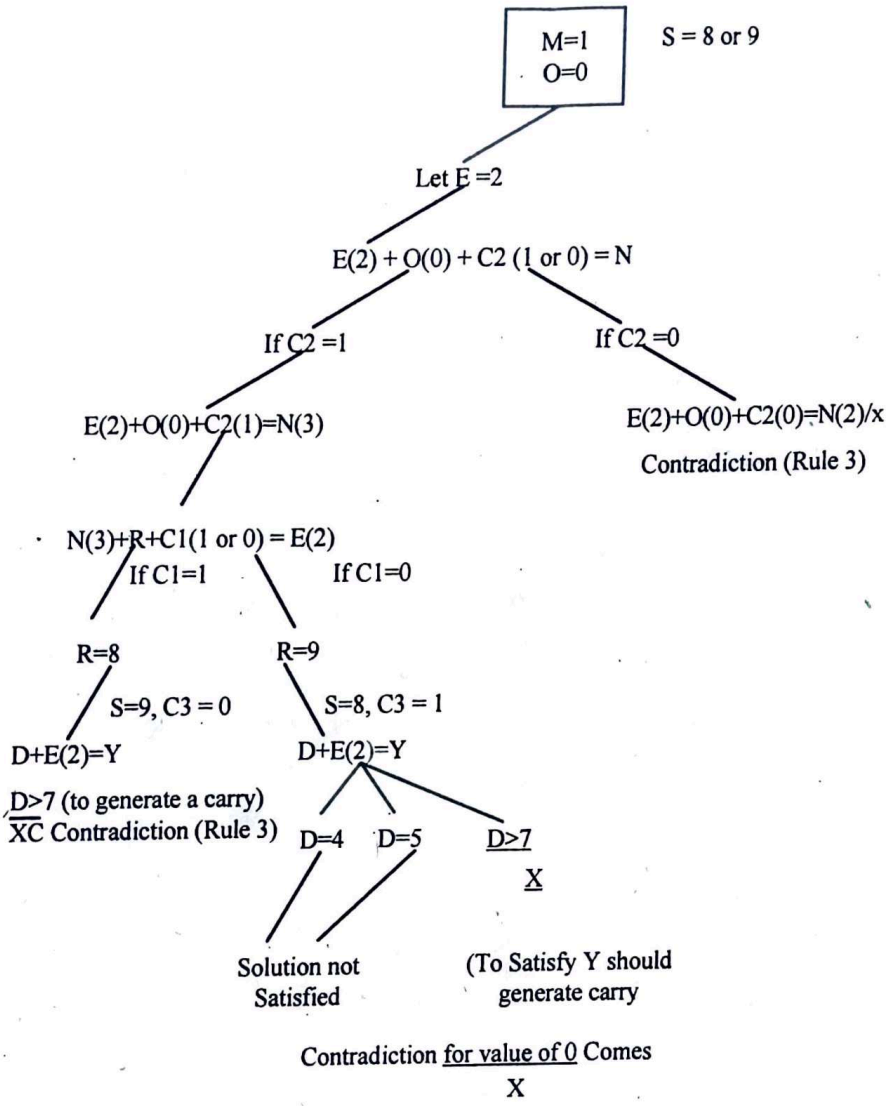
Goal State: the digits to the letters must be assigned in such a manner so that the sum is satisfied.

We are following the depth-first method to solve the problem.

1. initial guess m=1 because the sum of two single digits can generate at most a carry '1'.
2. When N=1 O=0 or 1 because the largest single digit number added to M=1 can generate the sum of either 0 or 1 depend on the carry received from the carry sum. By this we conclude that O=0 because m is already 1 hence we cannot assign same digit another letter (rule no.)
3. We have M=1 and O=0 to get O=0 we have S=8 or 9, again depending on the carry received from the earlier sum.

The same process can be repeated further. The problem must be composed into various constraints. And each constraint is to be satisfied by guessing the possible digits that the letters can be assumed that the initial guess has been already made. Rest of the process is being shown in the form of a tree in the figure below, using depth-first search for the clear understandability of the solution process.

Step-1



After Step 1 we derive are more conclusion that Y contradiction should generate a Carry. That is D+2>9

At step (4) we have assigned a single digit to every letter in accordance with the constraints and production rules.
Now by backtracking, we find the different digits assigned to different letters and hence reach the solution state.

Solution State:
Y=2
D=7
S=9
R=8
N=6
E=5
O=0
M=1
C1=1
C2=0
C3=0

	C3(0)	C2(1)	C1(1)	
	S(9)	E(5)	N(6)	D(7)
+	M(1),	O(0)	R(8)	E(5)
M(1)	O(0)	N(6)	E(5)	Y(2)

20. Write short notes on the following:
a) Constraint satisfaction problems **[WBUT 2009, 2011]**
b) Simulated Annealing **[WBUT 2013, 2015]**
c) Heuristic Search **[WBUT 2013]**
d) A* search **[WBUT 2018]**

Answer:

a) Constraint satisfaction problems:
Refer to Question No. 13(a) (1st Part) of Long Answer Type Questions.

b) Simulated Annealing:
Refer to Question No. 10 of Short Answer Type Questions.

c) Heuristic Search: ***Refer to Question No. 6 of Short Answer Type Questions.***

d) A* search: ***Refer to Question No. 20 of Long Answer Type Questions.***