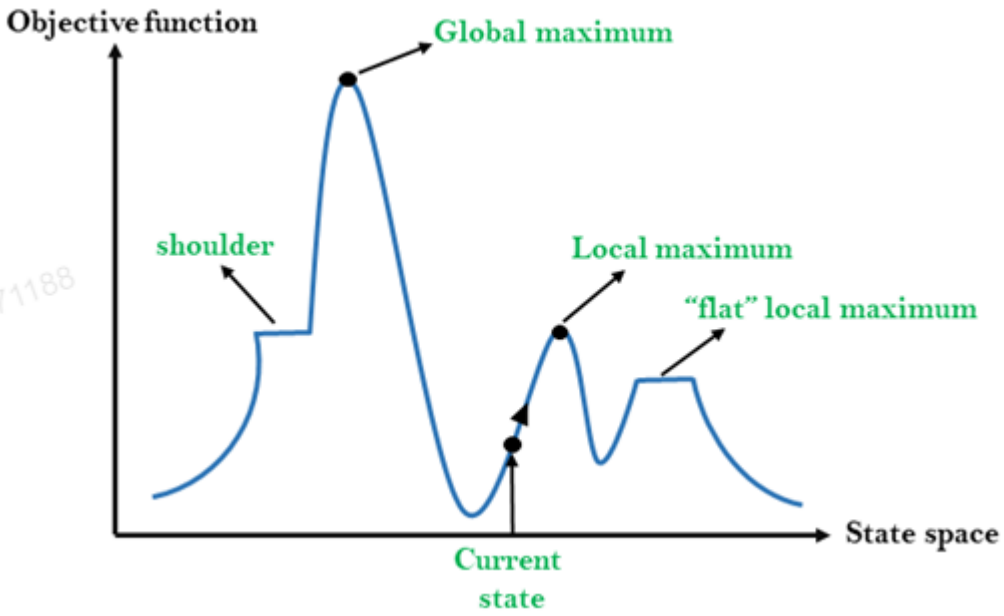
Module 3 - Problem solving

Explain Hill Climbing Algorithm also Explain What are the limitations of Hill Climbing Search and how that can be overcome?

Hill Climbing Algorithm is a local search algorithm used for optimization problems. It is called "hill climbing" because the algorithm tries to climb uphill in the search space by selecting the best available option at each step.

1) In hill climbing the test function is provided with a heuristic function which provides an estimate of how close a given state is to goal state.
2) Hill climbing is a mathematical optimization technique which belongs to the family of local search. It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution.
3) If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.
4) For example, hill climbing can be applied to the travelling salesman problem. It is easy to find an initial solution that visits all the cities but will be very poor compared to the optimal solution.
5) The algorithm starts with such a solution and makes small improvements to it, such as switching the order in which two cities are visited.
6) A node of hill climbing algorithm has two components which are state and value.
7) Hill Climbing is mostly used when a good heuristic is available.
8) In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state

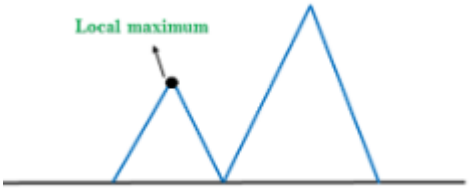


Different regions in the state space landscape:
1) Local Maximum: Local maximum is a state which is better than its neighbour states, but there is also another state which is higher than it.
2) Global Maximum: Global maximum is the best possible state of state space landscape. It has the highest value of objective function.
3) Current state: It is a state in a landscape diagram where an agent is currently present.
4) Flat local maximum: It is a flat space in the landscape where all the neighbour states of current states have the same value. Shoulder: It is a plateau region which has an uphill edge.

What are the limitations of Hill climbing? And give solutions on problems in Hill Climbing.

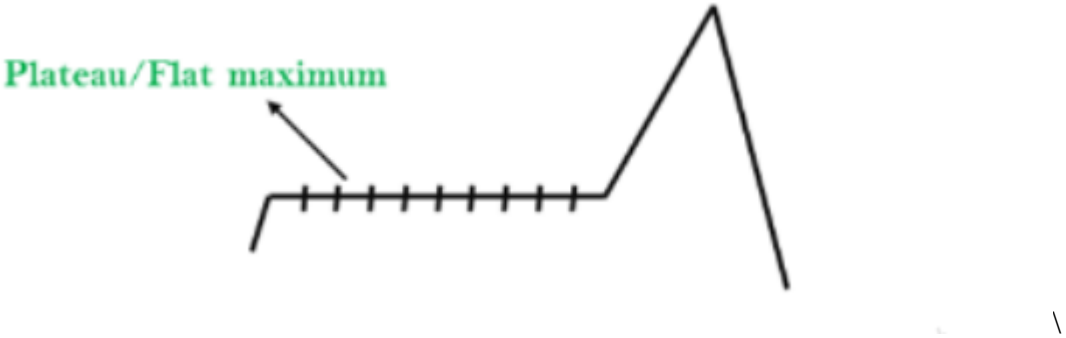
1. **Local Maximum:** A local maximum is a peak state in the landscape which is better than each of its neighbouring states, but there is another state also present which is higher than the local maximum.

Solution: Backtracking technique can be a solution of the local maximum in state space landscape. Create a list of the promising path so that the algorithm can backtrack the search space and explore other paths as well.



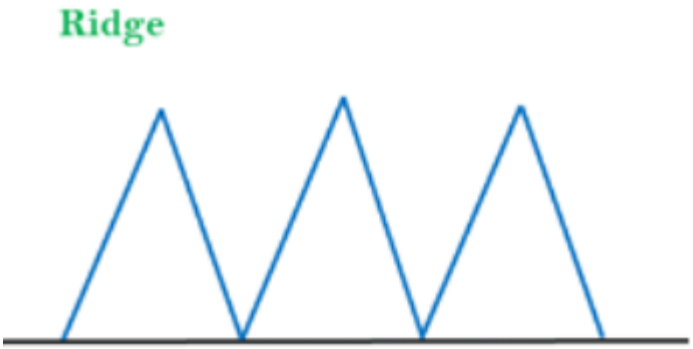
2. Plateau: A plateau is the flat area of the search space in which all the neighbour states of the current state contain the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area.

Solution: The solution for the plateau is to take big steps or very little steps while searching, to solve the problem. Randomly select a state which is far away from the current state so it is possible that the algorithm could find non-plateau region.



3. Ridges: A ridge is a special form of the local maximum. It has an area which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move.

Solution: With the use of bidirectional search, or by moving in different directions, we can improve this problem.



Overcoming the limitations of Hill Climbing Search:
To overcome these limitations, several variations of the hill climbing algorithm have been proposed, such as:
1. Random Restart: The algorithm can be restarted from a different initial solution to avoid getting stuck in a local maximum.
2. Simulated Annealing: This is a modification of the hill climbing algorithm that allows the algorithm to escape local maxima by accepting worse solutions with a certain probability.
3. Tabu Search: This variation of hill climbing uses a memory structure to avoid revisiting previously explored solutions.
4. Genetic Algorithms: Genetic algorithms use a population of solutions and apply mutation and crossover operators to generate new solutions, which can help overcome local maxima.
Overall, the hill climbing algorithm is a simple and intuitive optimization algorithm, but its effectiveness can be limited in some cases. By using modifications or other optimization algorithms, it is possible to improve its performance and overcome its limitations

Simulated Annealing with suitable examples:

Simulated Annealing is a metaheuristic optimization algorithm that is inspired by the physical process of annealing. Here's a pointwise explanation of simulated annealing with suitable examples:
1. Simulated Annealing is used to solve optimization problems, where the goal is to find the best solution among a set of possible solutions.
2. Simulated Annealing is based on the concept of annealing in metallurgy, where metals are heated and then slowly cooled to remove defects and improve their properties.
3. In simulated annealing, the process begins with an initial solution, and the algorithm tries to find a better solution by iteratively modifying the current solution.
4. During each iteration, the algorithm randomly generates a new candidate solution and evaluates its quality.
5. If the candidate solution is better than the current solution, it is accepted as the new current solution.
6. If the candidate solution is worse than the current solution, it is accepted with a certain probability, which decreases over time.
7. This probabilistic acceptance of worse solutions allows the algorithm to escape from local optima and explore the search space more thoroughly.
8. Simulated Annealing has been used in a variety of applications, such as circuit design, scheduling problems, and protein folding.

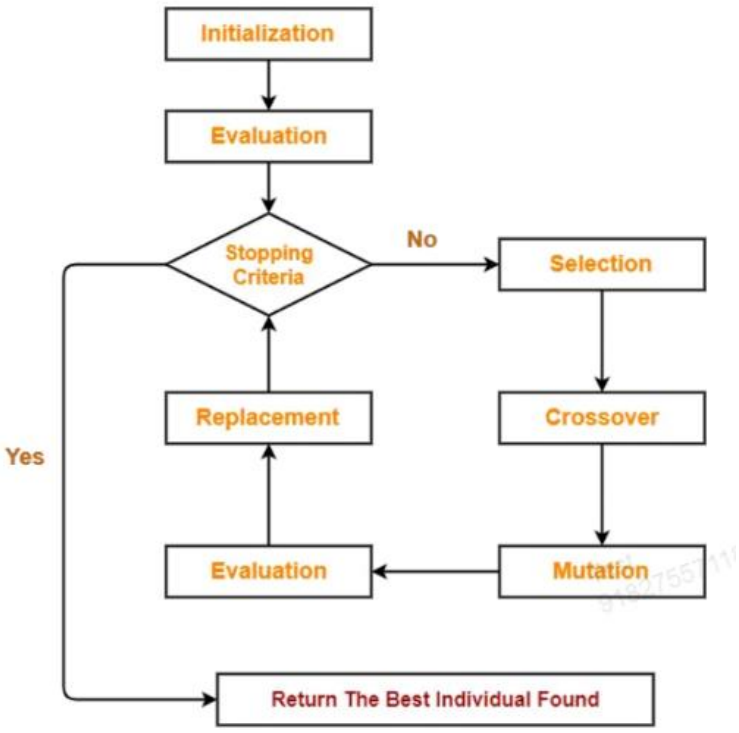
Example: Suppose you want to find the shortest route to travel through all the cities in a given set. This problem is known as the traveling salesman problem. Simulated Annealing can be used to find the optimal route.

1. First, you start with an initial solution, such as a random route through all the cities.
2. The algorithm generates a new candidate solution by randomly swapping two cities in the route.
3. The quality of the candidate solution is evaluated by computing the length of the route.
4. If the candidate solution is better than the current solution, it is accepted as the new current solution.
5. If the candidate solution is worse than the current solution, it is accepted with a certain probability, which decreases over time.
6. The algorithm repeats this process, gradually reducing the probability of accepting worse solutions over time.
7. Eventually, the algorithm converges to an optimal solution, which represents the shortest possible route through all the cities.

To summarize, Simulated Annealing is an optimization algorithm that iteratively searches for the best solution among a set of possible solutions by gradually modifying the current solution and accepting worse solutions with a certain probability. It is useful in solving complex optimization problems where traditional optimization techniques are not feasible

Explain Genetic Algorithm [chromosome, Mutation, Cross Over]

Genetic algorithm is a type of optimization algorithm that mimics the process of natural selection to evolve a population of candidate solutions to an optimization problem. It uses techniques such as selection, crossover, and mutation to evolve better and better solutions over time. Here's a point wise explanation of genetic algorithm



How Genetic Algorithm Works

1. Initialization: The algorithm starts by generating an initial population of random candidate solutions. Each solution is represented by a chromosome, which is a string of genes that encode the values of the decision variables in the problem.

2. Fitness function: The fitness function is a function that evaluates the quality of each candidate solution based on the objective function of the optimization problem. The fitness function assigns a fitness score to each solution, which is used to determine the selection probability of each individual in the population.

3. Selection: The selection process involves choosing the fittest individuals from the current population to be parents for the next generation. Selection is typically done using a fitness-proportionate or rank-based selection scheme, in which individuals with higher fitness scores are more likely to be selected.

4. Crossover: Crossover is the process of combining two parent chromosomes to create a new offspring chromosome. This is done by selecting a crossover point in the parent chromosomes and exchanging the genes to the right of that point between the two parents. This creates two new offspring chromosomes that are a combination of the parent chromosomes.

5. Mutation: Mutation is the process of randomly changing a gene in a chromosome to create a new candidate solution. This is done to introduce new variations in the population and prevent the algorithm from getting stuck in local optima.

6. Reproduction: The reproduction process involves creating a new population of candidate solutions by combining the selected individuals using crossover and mutation. The new population is then evaluated using the fitness function to determine the fittest individuals in the new population.

7. Termination: The algorithm terminates when a stopping criterion is met, such as a maximum number of generations or a satisfactory fitness score. The best candidate solution found by the algorithm is then returned as the solution to the optimization problem.

To summarize, genetic algorithm is a type of optimization algorithm that uses techniques such as selection, crossover, and mutation to evolve a population of candidate solutions to an optimization problem. The algorithm starts by generating an initial population of random solutions and then evolves better and better solutions over time through selection, crossover, and mutation. The algorithm terminates when a stopping criterion is met, and the best candidate solution found by the algorithm is returned as the solution to the optimization problem.

A* Algorithm with an Example [VV IMP]

The A* algorithm is a popular pathfinding algorithm used in artificial intelligence and robotics to find the shortest path between two points. It uses a heuristic function to guide the search towards the goal node.

● A* Algorithm is one of the best and popular techniques used for path finding and graph traversals.

● A lot of games and web-based maps use this algorithm for finding the shortest path efficiently.

● It is essentially a best first search algorithm.

Working :

A* Algorithm works as-
● It maintains a tree of paths originating at the start node.
● It extends those paths one edge at a time.
● It continues until its termination criterion is satisfied.

A* Algorithm extends the path that minimizes the following function-

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

Here,
● ‘n’ is the last node on the path
● g(n) is the cost of the path from start node to node ‘n’
● h(n) is a heuristic function that estimates cost of the cheapest path from node ‘n’ to the goal node

Algorithm :
1. Add the starting node to the open list.
2. While the open list is not empty, select the node with the lowest f-score.
3. If the current node is the goal node, return the path.
4. For each neighbour of the current node, calculate its tentative g-score and f-score.
5. If the neighbour is not in the open list, add it.
6. Otherwise, update the neighbour’s g-score and f-score if the tentative g-score is lower.

7. Repeat steps 2 to 6 until the goal node is found or the open list is empty.
8. If the open list is empty and the goal node has not been found, return failure.

Example

Given an initial state of a 8-puzzle problem and final state to be reached-

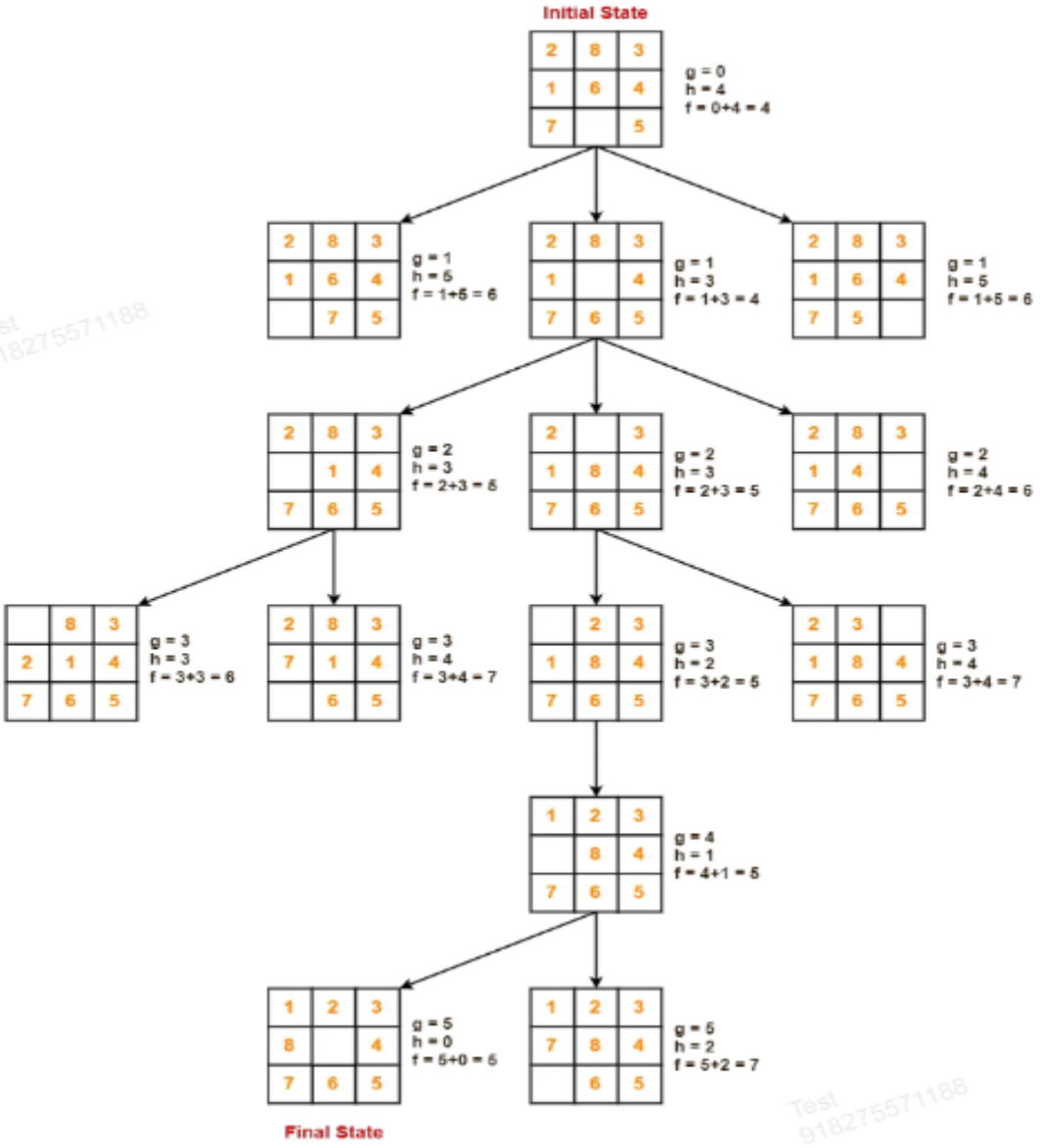
Initial State

Final State

Find the most cost-effective path to reach the final state from initial state using A* Algorithm.
Consider g(n) = Depth of node and h(n) = Number of misplaced tiles.

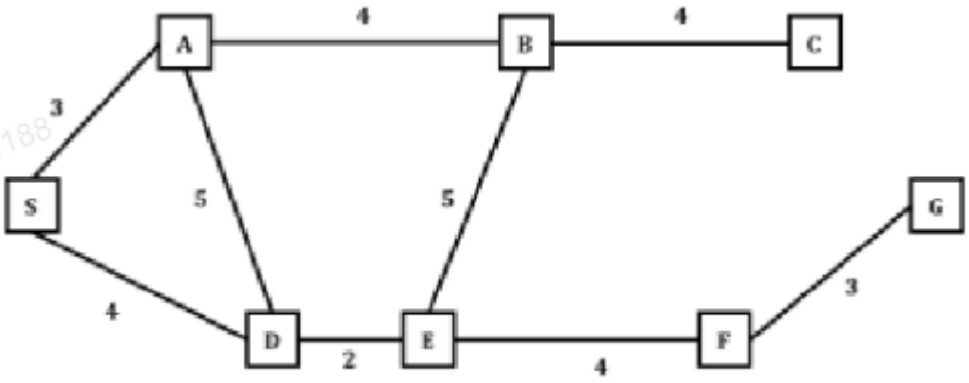
Solution-

1. A* Algorithm maintains a tree of paths originating at the initial state.
2. It extends those paths one edge at a time.
3. It continues until final state is reached.



Question . Apply A* algorithm on the following graph. Heuristic values are h(S) = 15, h(A) = 14, h(D) =12, h(B) = 10, h(E) = 10, h(C) = 8, h(F) = 10 , h(G) = 0.

S is the start node and G is the goal node.



We start with Note S
X = S
where X is the Current Node

- Node A and D can be reached from Node S
- Open (A , D)
- Lets Calculate f(S-A) and f(S-D)
- f(S-A) = g(n) + h(n) = 3 + 14 = 17
- f(S-D) = 4 + 12 = 16
- Since f(S-D) is least it decided to go to Node D

Now X = D and Close (S)

- Node A and E can be Reached from Node D
- Open (A , E)
- Lets Calculate f(S-D-A) and f(S-D-E)
- f(S-D-A) = g(n) + h(n) = 9 + 14 = 23
- f(S-D-E) = 6 + 10 = 16
- Since f(S-D-E) is least it decided to go to Node E

Now X = E and Close (S,D)
- Node B and F can be Reached from Node E
- Open (B , F)
- Lets Calculate f(S-D-E-B) and f(S-D-E-F)
- f(S-D-E-B) = g(n) + h(n) = 11 + 10 = 21
- f(S-D-E-F) = 10 + 10 = 20
- Since f(S-D-E-F) is least it decided to go to Node F

Now X = F and Close (S,D,E)
- Node G can be Reached from Node F
- Open (G)
- Lets Calculate f(S-D-E-F-G)
- f(S-D-E-F-G) = g(n) + h(n) = 13 + 0 = 13

we reached the goal node
Path : S -> D -> E -> F -> G
This is the required Shortest Path from Node S to Node G

Q. Explain Alpha Beta Pruning Algorithm ?

○ Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.

○ As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called pruning. This involves two threshold parameter Alpha and beta for future expansion, so it is called alpha-beta pruning. It is also called as Alpha-Beta Algorithm.

Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.

○ The two-parameter can be defined as:

a. Alpha: The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -∞.

b. Beta: The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is +∞.

○ The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

Condition for Alpha-beta pruning:

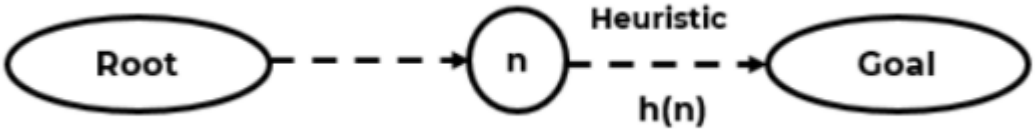
The main condition which required for alpha-beta pruning is:

$$\alpha >= \beta$$

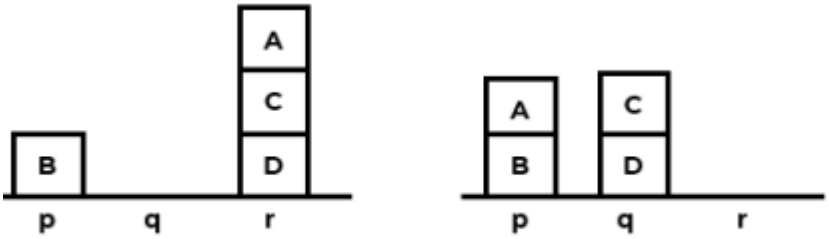
Key points about alpha-beta pruning:
○ The Max player will only update the value of alpha.
○ The Min player will only update the value of beta.
○ While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
○ We will only pass the alpha, beta values to the child nodes.

Define heuristic function. Give an example heuristics function for Blocks World Problem

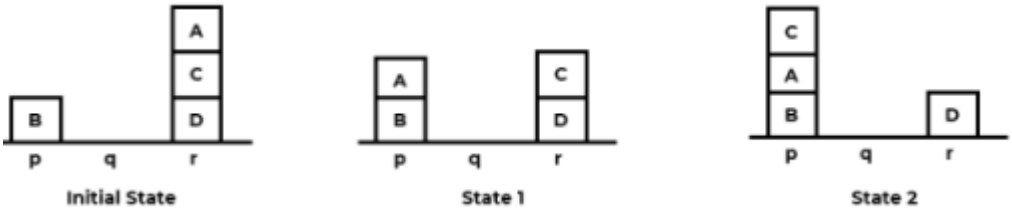
HEURISTIC FUNCTION:
1. Heuristic Function is the function that gives an estimation on the Cost of getting from node to the goal
state.
2. Heuristic Function is used in Informed Search Technique.
3. It is used in a decision process to try to make the best choice of a list of possibilities.
4. Best move is the one with the least cost.
5. Heuristic function helps in implementing goal oriented search.
6. It is usually used to increase the efficiency of the search process



EXAMPLE OF HEURISTICS FUNCTION FOR BLOCKS WORLD PROBLEM



As shown below the heuristic value of H(s) = 4. Since we take 4 move to reach from initial state to Goal State



Define heuristic function. Give an example of a heuristics function for an 8-puzzle problem.

Definition is same as above now Let's take an example of heuristics function for an 8-puzzle problem.

HEURISTICS FUNCTION FOR 8-PUZZLE PROBLEM:

For the 8-Puzzle Problem, two heuristics are commonly used. i.e. No. of misplaced tiles and Manhattan Distance.

No. of Misplaced tiles:
- In this case, heuristic function is determined using number of misplaced titles (not including the blank space)

- For Example:
Consider the example given below. It consists of initial state and goal state

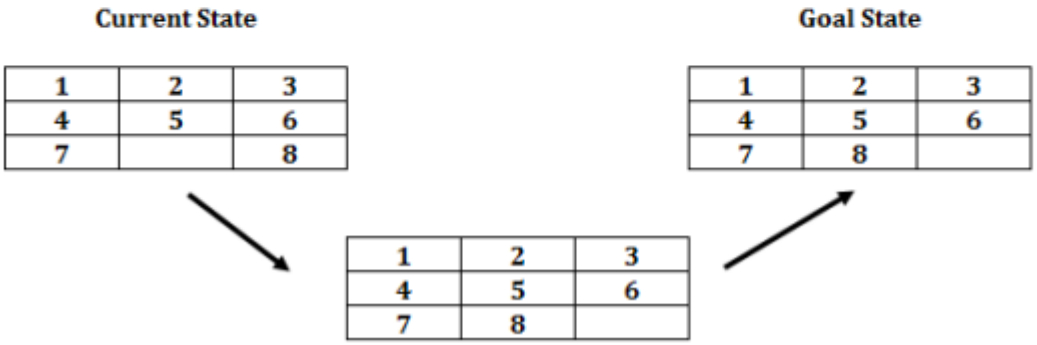
Current State

1	2	3
4	5	6
7		8

Goal State

1	2	3
4	5	6
7	8	

s shown in example, only "8" is misplaced, so the heuristic function evaluates to 1.



∴ Heuristic h (n) = 1

II) Manhattan Distance:

1. Manhattan Distance is the sum of the distances of the tiles from their goal positions.
For Example:

- Consider the example given below. It consists of initial state and goal state.
- In this case, only the "3", "8" and "1" titles are misplaced by 2, 3 and 3 spaces respectively. So the heuristic function evaluates to 8.

- Therefore, Heuristic h (n) = 2 + 3 + 3 = 8

Current State

3	2	8
4	5	6
7	1	

2 Spaces

3	→	3

3 Spaces

	←	8
	↓	
	8	

Goal State

1	2	3
4	5	6
7	8	

3 Spaces

1	←	
	↑	
	1	