AI & NN
Unit I – Part 3

A* Search Algorithm, Hill Climbing Algorithm

Class: III CYS

Sem : 2025-26, Sem 5

-Dr. S.Durga, M.E., PhD

A* Search Algorithm : Introduction

- The A* search algorithm is a widely used pathfinding algorithm in artificial intelligence
- Is an informed search algorithm
- It efficiently finds the most cost-effective path between a starting point and a goal by intelligently exploring a graph or search space.
- It is known for its balance between efficiency and optimality, often outperforming simpler algorithms like Dijkstra's or Breadth-First Search by using a heuristic to estimate the distance to the goal.

A* Search Algorithm: Key concepts

- **Heuristic Function:** A* utilizes a heuristic function (h(n)) that estimates the cost from a given node (n) to the goal. This helps guide the search towards promising paths.
- **Evaluation Function:** The algorithm uses a combined evaluation function (f(n) = g(n) + h(n)), where g(n) is the cost to reach node n from the start and h(n) is the heuristic estimate.
- **Optimality and Completeness:** A* is guaranteed to find the shortest path (optimal) if the heuristic function is admissible (never overestimates the actual cost).
- **Efficiency:** A* is generally more efficient than simpler algorithms because it prioritizes exploring nodes that are likely to lead to the goal, saving computational resources.

How A* Works:

1. Initialization:

- Start node is added to an open list (nodes to be evaluated).
- The starting node's g(n) is set to 0 and h(n) is calculated.

2. Iteration:

- The node with the lowest f(n) value is selected from the open list.
- If it's the goal node, the path is found, and the algorithm terminates.
- Otherwise, the node is expanded (its neighbors are examined).

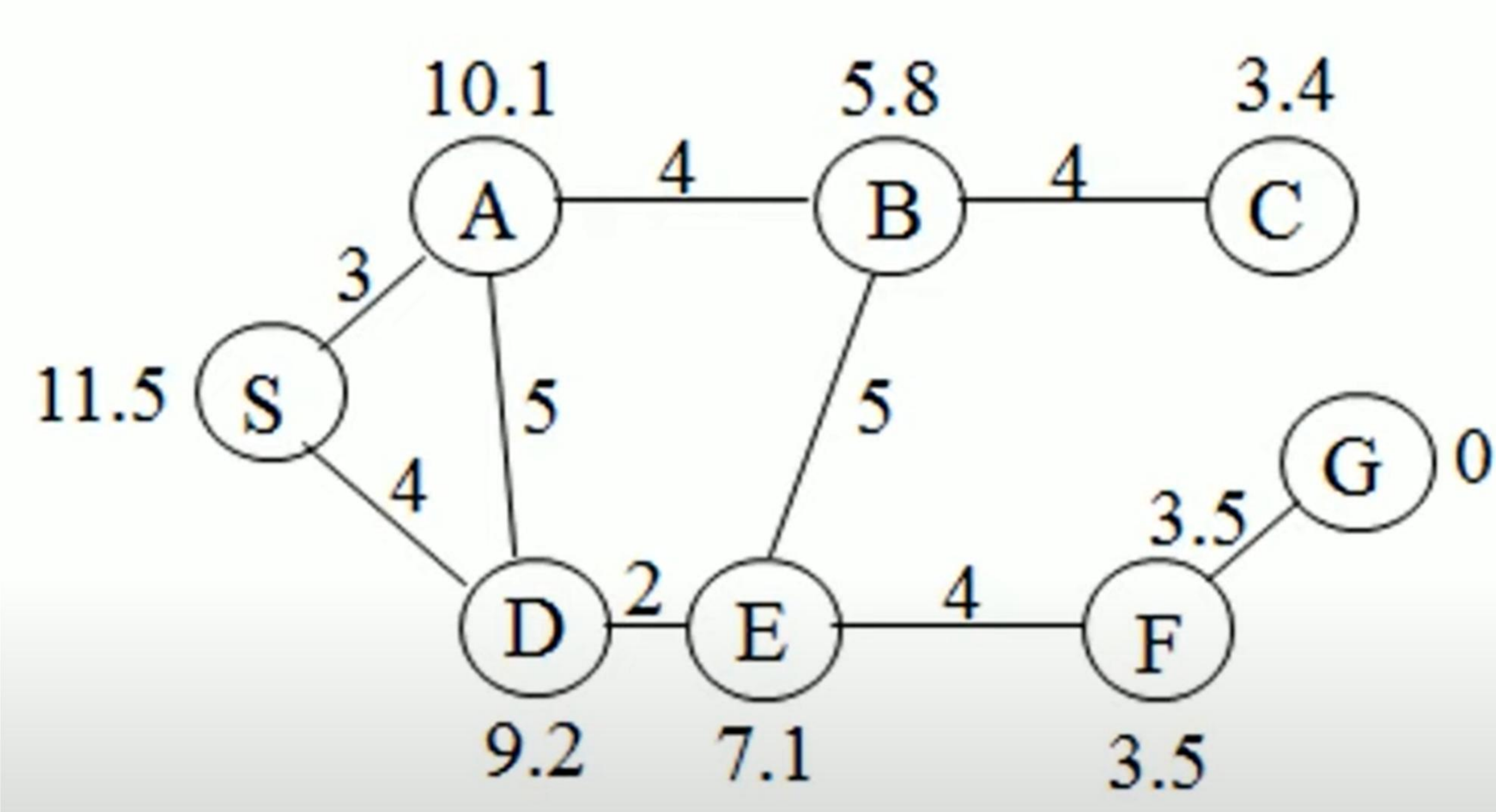
3. Neighbor Evaluation:

- For each neighbor, the g(n), h(n), and f(n) values are calculated.
- If the neighbor is not already in the open list or a closed list (nodes already evaluated), it is added to the open list.
- If the neighbor is already in the open or closed list, and the new path to it is shorter, update the path and its associated costs.

4. Repeat:

- The process repeats until the goal is found or the open list is empty (no solution).

Ex. Find shortest path from S to G

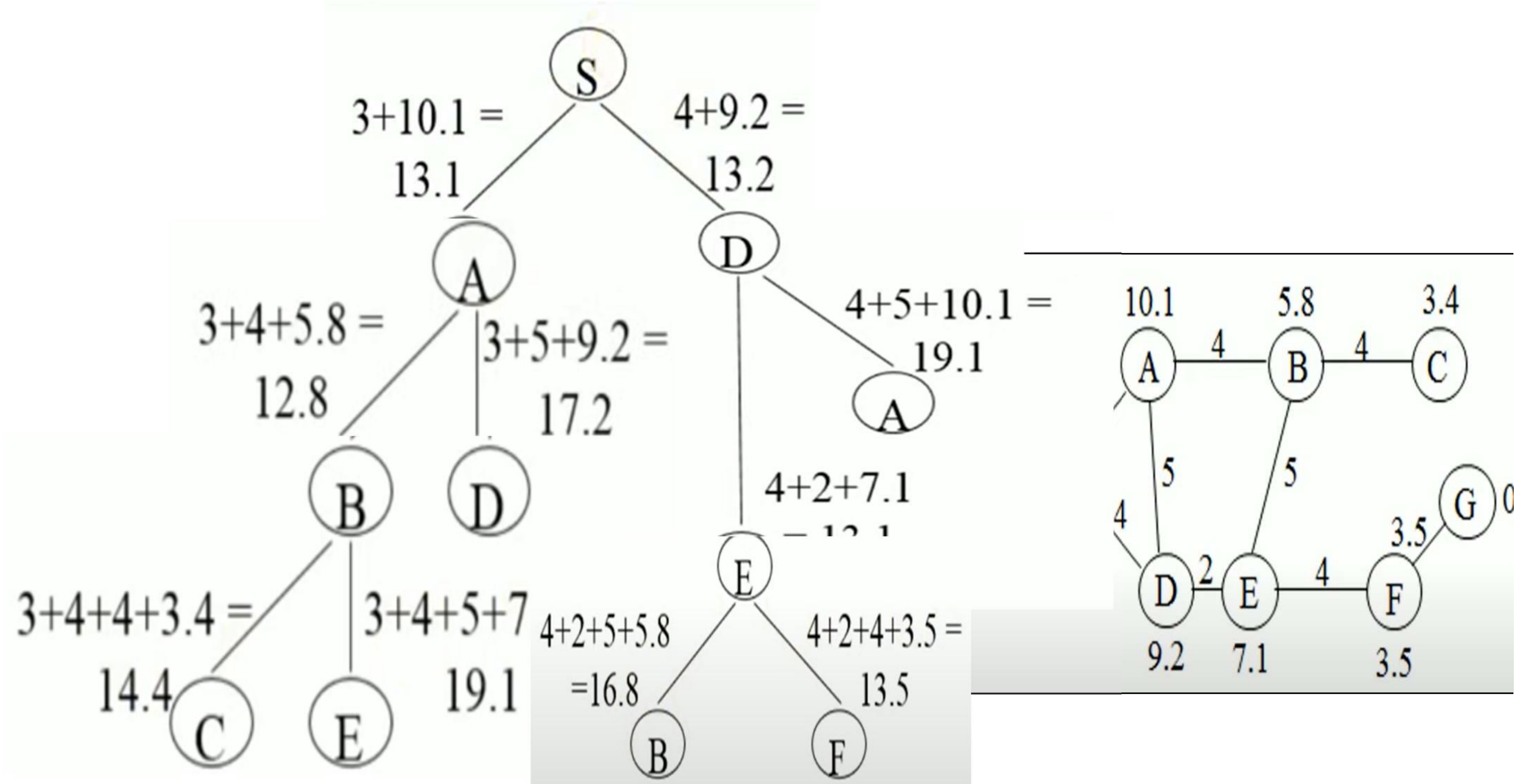


For each successor node, we need to calculate

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

$h(n)$ = cost of the cheapest path from node n to a goal state.

$g(n)$ = cost of the cheapest path from the initial state to node n.


S
3+10.1 =
13.1
4+9.2 =
13.2
A
D
3+4+5.8 =
12.8
3+5+9.2 =
17.2
4+5+10.1 =
19.1
A
B
D
4+2+7.1
E
3+4+4+3.4 =
14.4
3+4+5+7
19.1
C
E
4+2+5+5.8
=16.8
4+2+4+3.5 =
13.5
B
F
10.1
5.8
3.4
A
4
B
4
C
5
5
4
G
0
3.5
D
2
E
4
F
9.2
7.1
3.5

Advantages:

- **Optimal Path:** Guarantees the shortest path if the heuristic is admissible.
- **Efficient Search:** Directs the search towards the goal, avoiding unnecessary exploration of less promising paths.

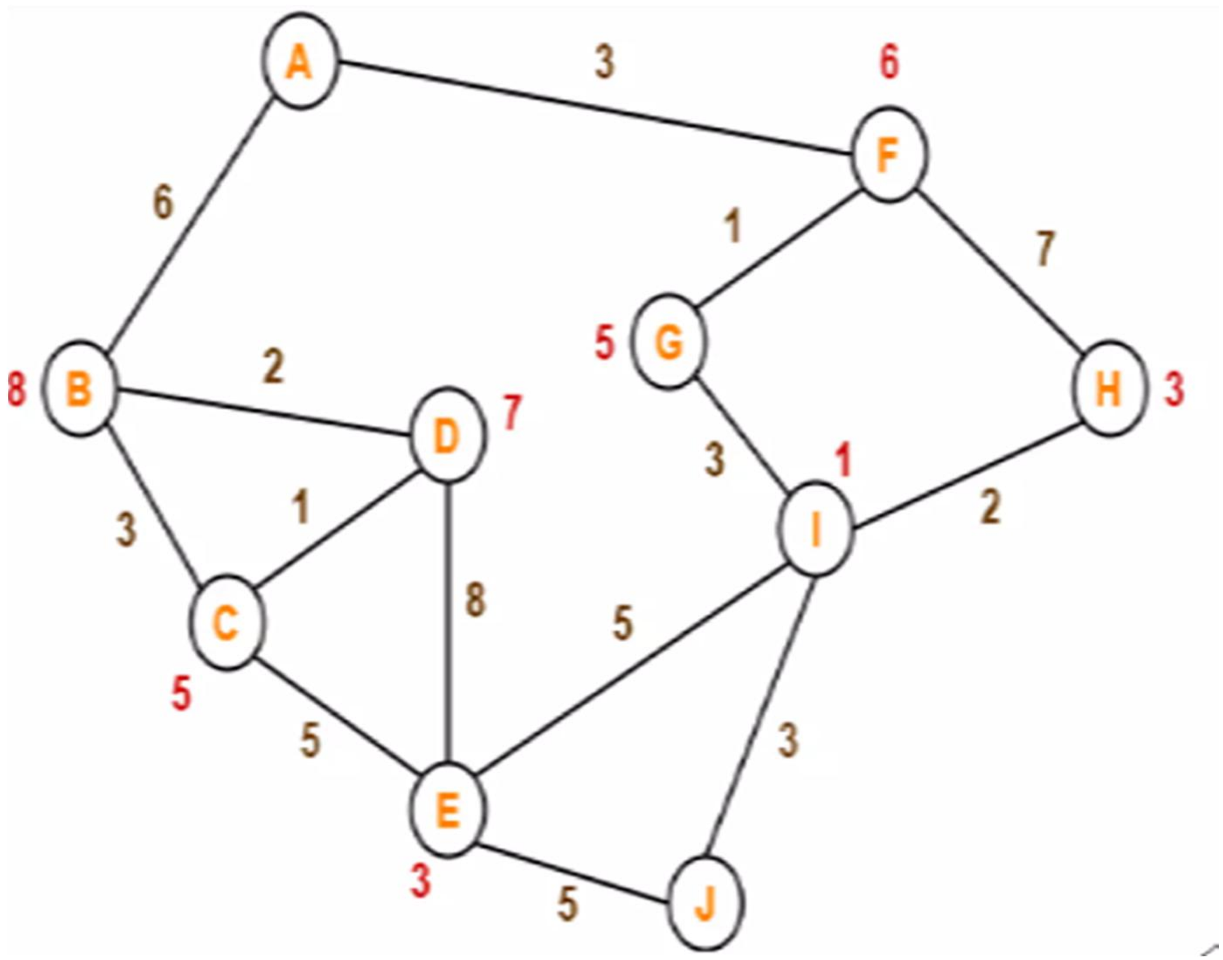
Disadvantages:

- **Memory Usage:** Can require significant memory to store the open and closed lists, especially in complex search spaces.

Applications of A* Search

- A* is widely used in various applications, including:
- **Robotics:** Path planning and navigation.
- **Video Games:** Character movement and AI pathfinding.
- **Navigation Systems:** GPS and route finding.

Homework : find path from ‘A’ to ‘J’



Hill climbing Algorithm

Hill climbing is a local search algorithm in artificial intelligence used to find the optimal solution to a problem by iteratively improving an initial solution.

It works by moving towards a better solution in the search space, similar to climbing a hill towards its peak, until a peak (optimal solution) is reached.

Hill Climbing as a Heuristic Search in Mathematical Optimization

- Hill Climbing algorithm often used for solving **mathematical optimization problems** in AI.
- With a good **heuristic function** and a large set of inputs, Hill Climbing can find a sufficiently good solution in a reasonable amount of time, although it may not always find the **global optimal maximum**.
- In mathematical optimization, Hill Climbing is commonly applied to problems that involve **maximizing or minimizing a real function**.
- For example, in the **Traveling Salesman Problem**, the objective is to minimize the distance traveled by the salesman while visiting multiple cities.

Steps

- Start with an initial solution.
- Evaluate neighboring solutions.
- Move to the neighbor with the best value (e.g., lowest cost or highest fitness).
- Repeat until no better neighbor is found, indicating a local optimum.

Key concept

Local Search: Hill climbing is a type of local search, meaning it explores the solution space by making small, incremental changes to the current solution.

Optimization Problem: It's used to solve optimization problems where the goal is to find the best solution (e.g., maximizing profit or minimizing cost).

Heuristic: Hill climbing is a heuristic algorithm, meaning it doesn't guarantee finding the absolute best solution (global optimum) but aims to find a good solution within a reasonable time.

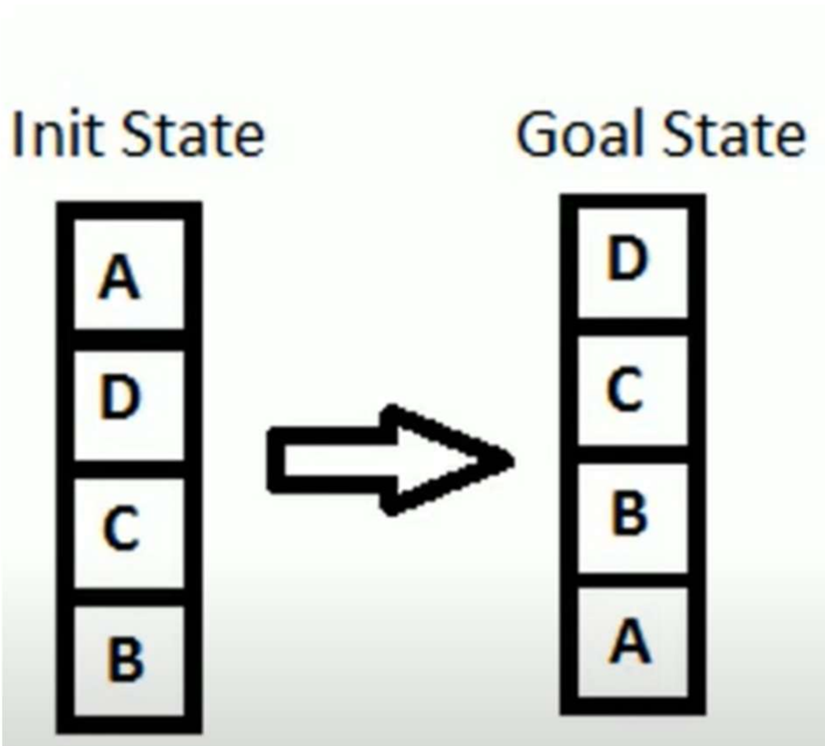
What is a Heuristic Function?

- A **heuristic function** is a function that ranks the possible alternatives at any branching step in a search algorithm based on available information.
- It helps the algorithm select the best route among various possible paths, thus guiding the search towards a good solution efficiently.

Types of Hill Climbing:

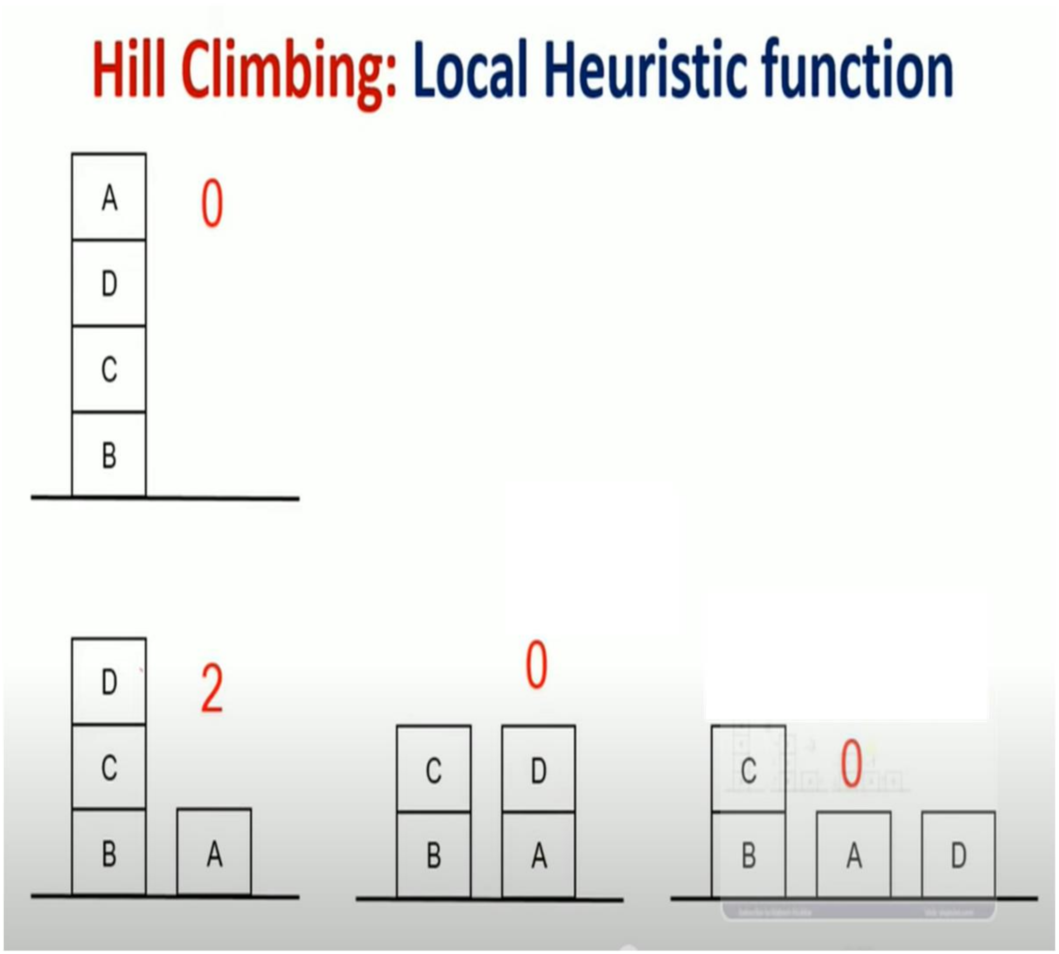
- **Simple Hill Climbing:** Examines only one neighbor at a time and moves to it if it's better than the current state.
- **Steepest Ascent/Descent:** Examines all neighbors and moves to the one that provides the greatest improvement.
- **Stochastic Hill Climbing:** Randomly selects a neighbor to explore, which can help avoid getting stuck in local optima.

Example



Two ways:

Local and global heuristic functions



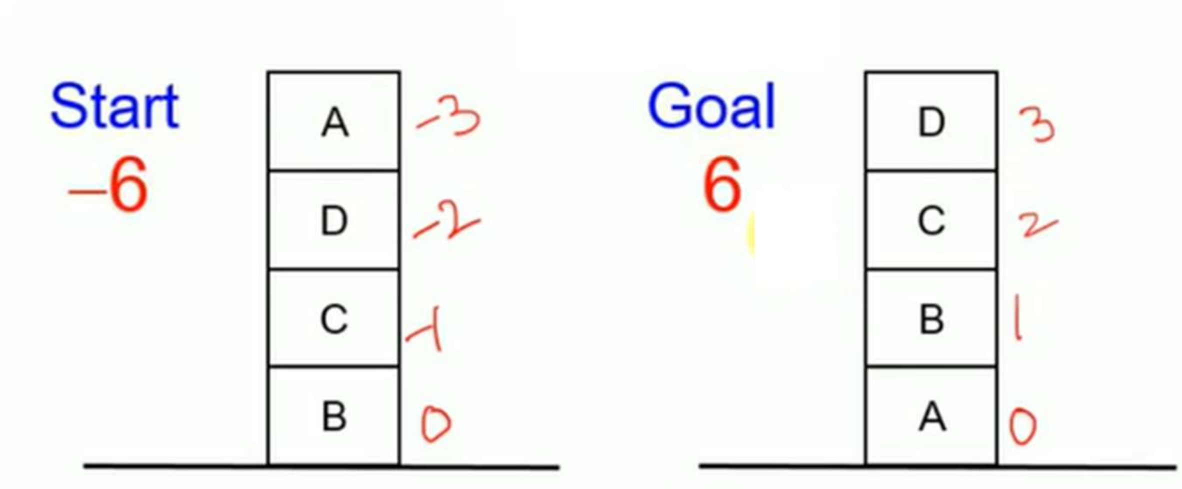
- Local heuristic:
- +1 for each block that is resting on the thing it is supposed to be resting on
- -1 for each block that is resting on a wrong thing

Goal not reached at local optimum

- There are two options: either to move D on to of A or to move D to the ground
- In both a cases we have obtained cost value 0.
- This point is called as local maximum or local optimum.
- Because of this we cant goahead from here onwards. This is a disadv. of using local elastic function.

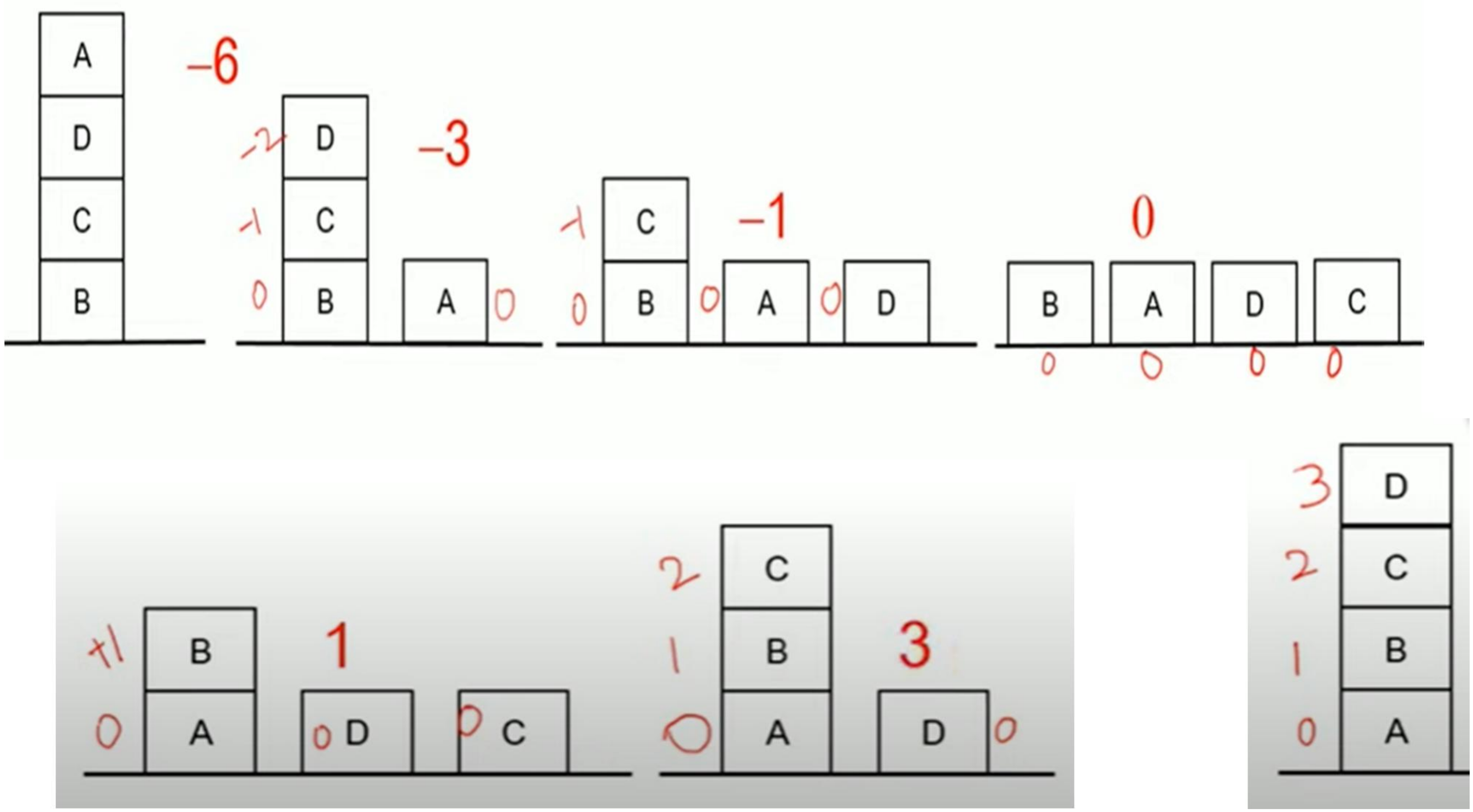
Option 2: using Global heuristic

- Global heuristic:
- For each block that has the correct support structure: +1 to every block in the support structure
- For each block that has wrong support structure: -1 to every block in the support structure



- Below B nothing is there so it will be assigned with value 0.
- Below C, only B is present which is not the correct support structure. Hence assign every block below C (in current state) with -1.
- Below D, there are two blocks C and B in the current state. But as per the Goal state three blocks such as C, B and A should be there in order. Hence assign every block below D with -1, add them and assign to D

Hill Climbing: Global Heuristic function



Final cost is 6 and is matched with goal

Limitations:

- **Local Optima:**
- Hill climbing can get stuck in local optima, which are solutions that are better than their neighbors but not the best overall solution.
- **No Backtracking:**
- It doesn't remember previous states, which can hinder finding the global optimum.

Applications:

- **Machine Learning:** Feature selection, hyperparameter tuning.
- **Traveling Salesperson Problem:** Finding a near-optimal route.
- **Game Playing:** Determining optimal moves in a game.

Algorithm for Simple Hill Climbing

1. Evaluate the initial state. If it is a goal state, return success.
2. Make the initial state the current state.
3. Loop until a solution is found or no operators can be applied:
    1. Select a new state that has not yet been applied to the current state.
    2. Evaluate the new state.
    3. If the new state is the goal, return success.
    4. If the new state improves upon the current state, make it the current state and continue.
    5. If it doesn't improve, continue searching neighboring states.
4. Exit the function if no better state is found.

Lab Activity

- **Implement a A* Algorithm using Python**
- **Implement a Simple Hill Climbing Algorithm in Python**