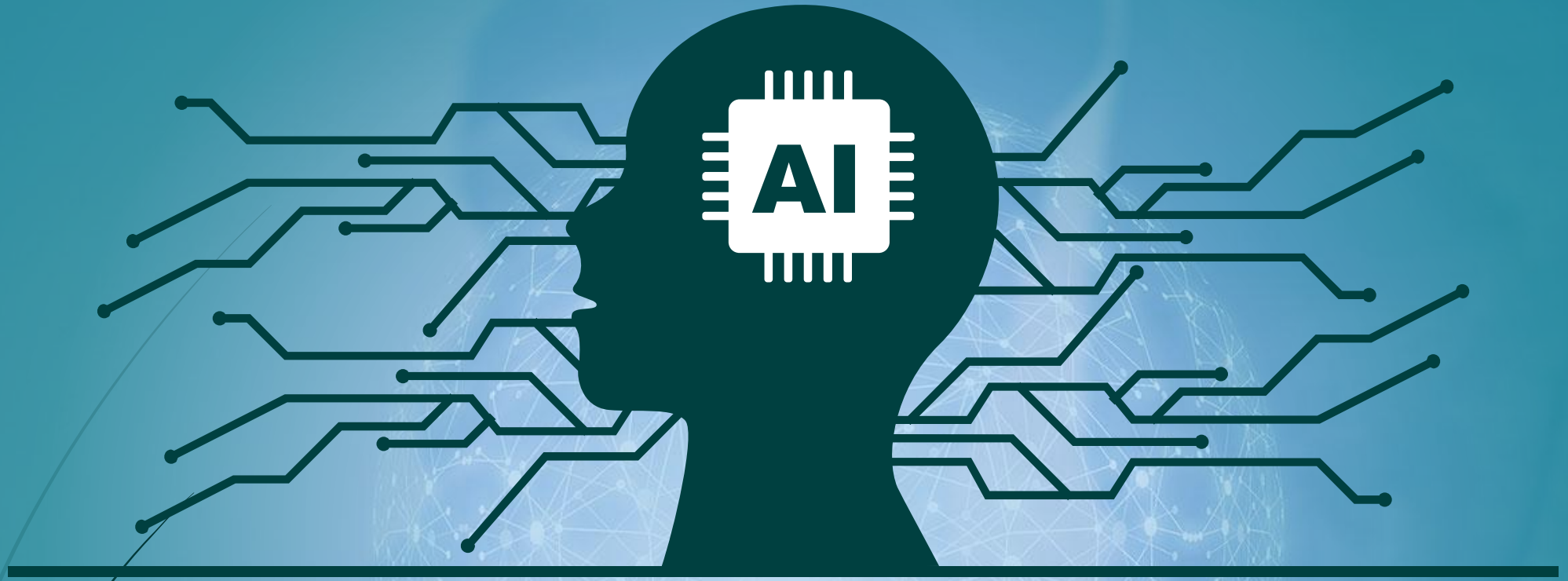


Advanced Artificial Intelligence

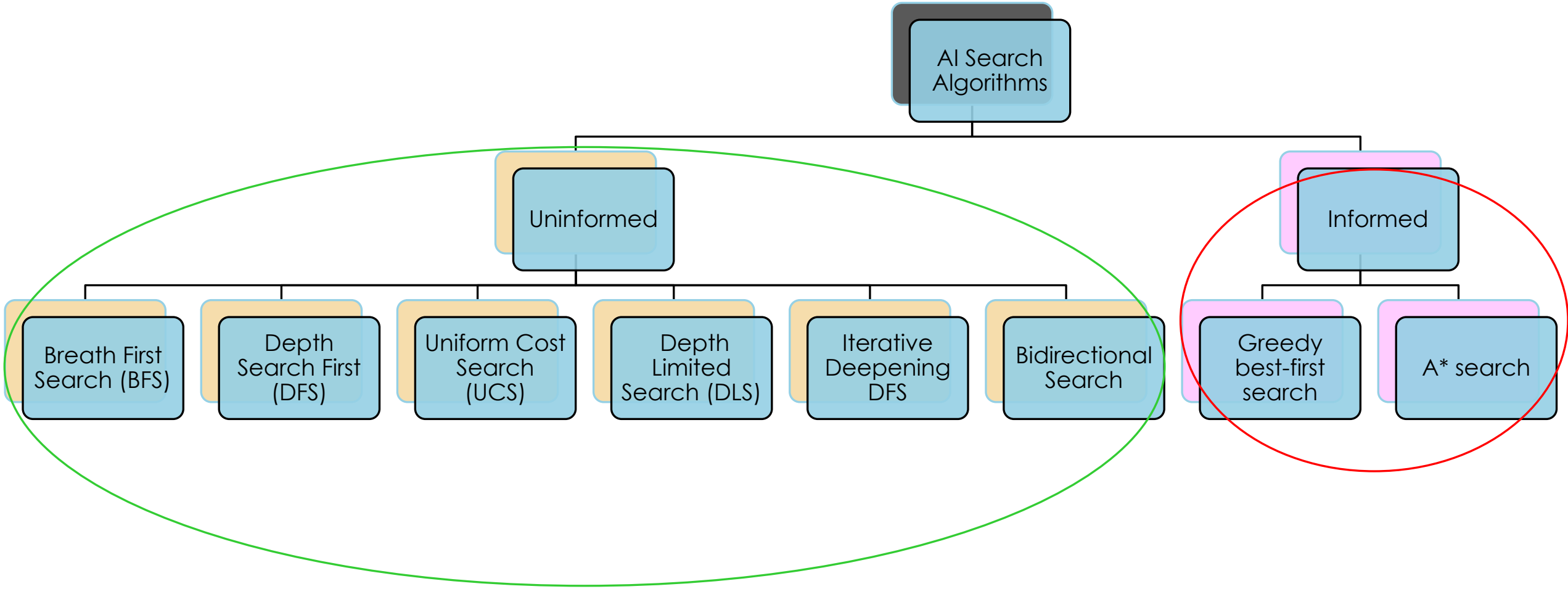
Week #4

Dr. Qurat Ul Ain
Assistant Professor
Dept. of AI & DS
FAST NUCES, Islamabad
Email:

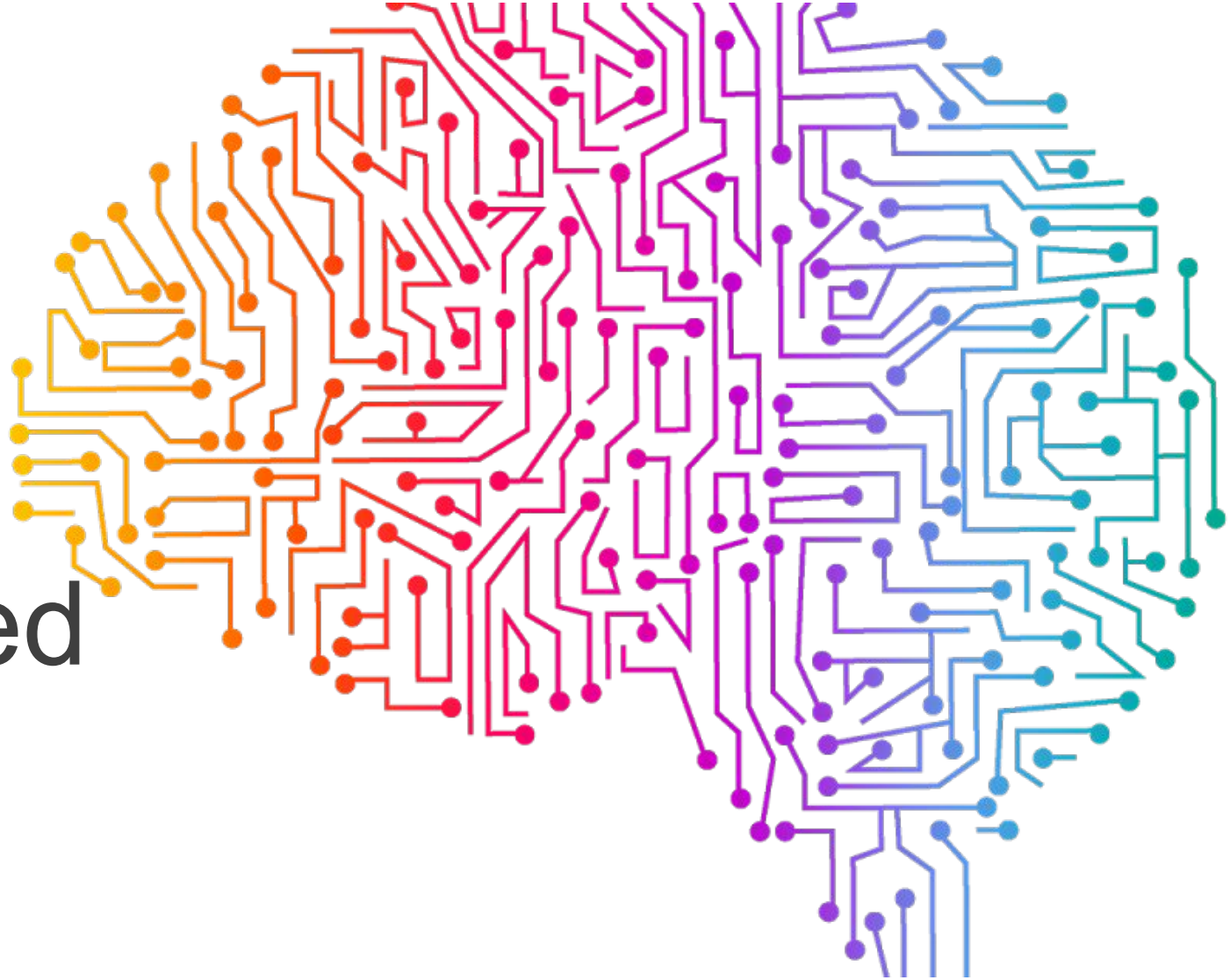
INFORMED SEARCH ALGORITHM

Learning Objective of this Topic

- What is Informed Search
- Informed Searching Algorithms
  - Best-first Search Algorithm (Greedy Search)
  - A* Search
- Advantages and Disadvantages of Informed Search
- Puzzle Problem by A* Search

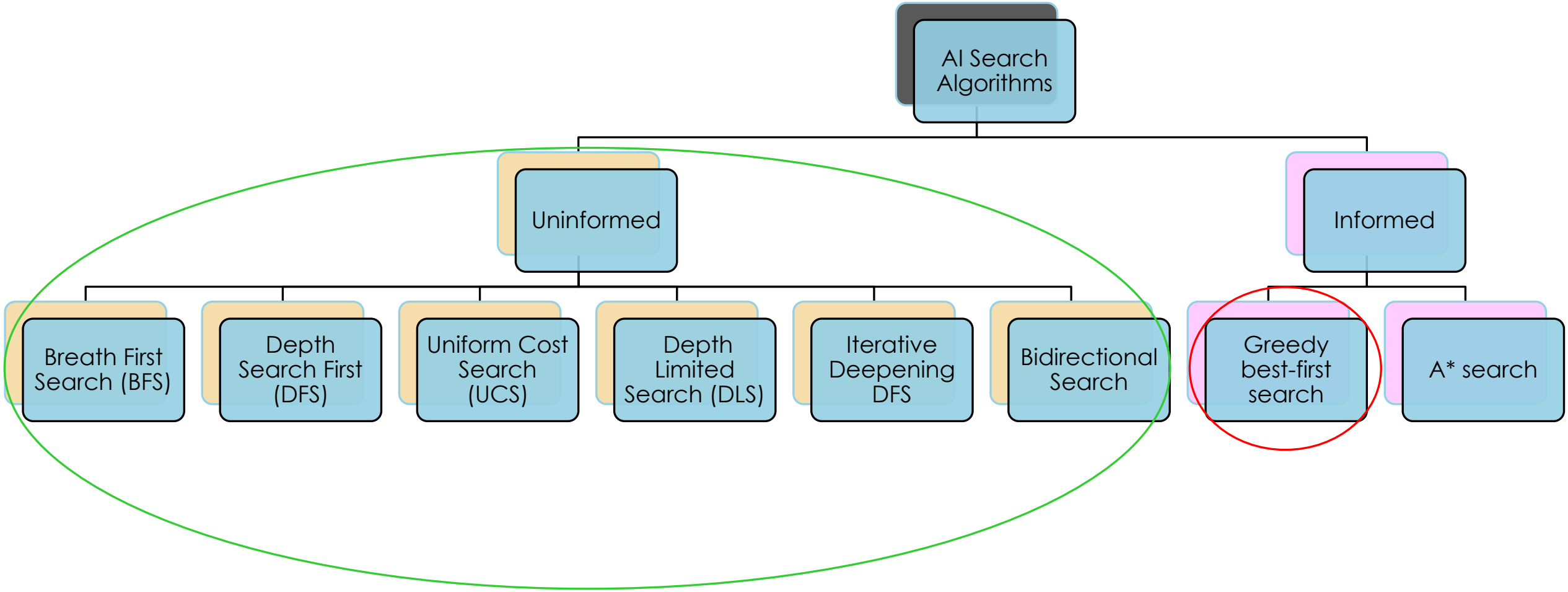

AI Search Algorithms
Uninformed
Informed
Breath First Search (BFS)
Depth Search First (DFS)
Uniform Cost Search (UCS)
Depth Limited Search (DLS)
Iterative Deepening DFS
Bidirectional Search
Greedy best-first search
A* search

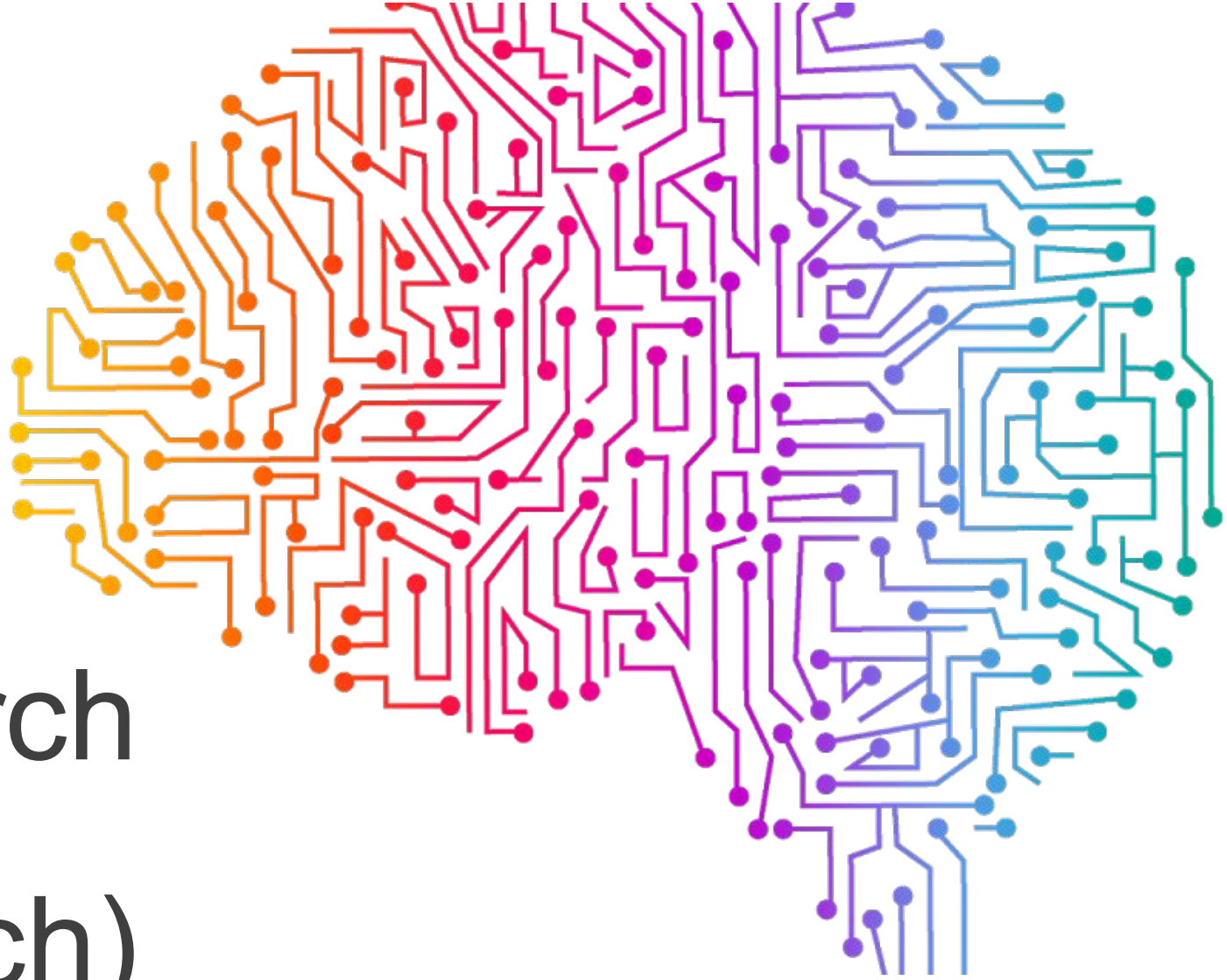
What is Informed Search?



Informed Search

- The informed search contains knowledge such as how far we are from the goal, path cost, how to reach to the goal node, etc.
- This knowledge helps agents explore less in the search space and find the goal node.
- An informed search algorithm uses the idea of **heuristic**, so it is also called Heuristic search.
- **Heuristic function:** It is a technique designed to solve a problem quickly.
  - In searching, it produces the estimation of how close the agent is to the goal.
- The heuristic approach, however, might not always give the best solution, but it is guaranteed to find a good solution in a reasonable time.
- It is represented by h(n), and the value of the heuristic function is always positive.


AI Search Algorithms
Uninformed
Informed
Breath First Search (BFS)
Depth Search First (DFS)
Uniform Cost Search (UCS)
Depth Limited Search (DLS)
Iterative Deepening DFS
Bidirectional Search
Greedy best-first search
A* search



Best-first Search

(Greedy Search)

Best-first Search Algorithm (Greedy Search)

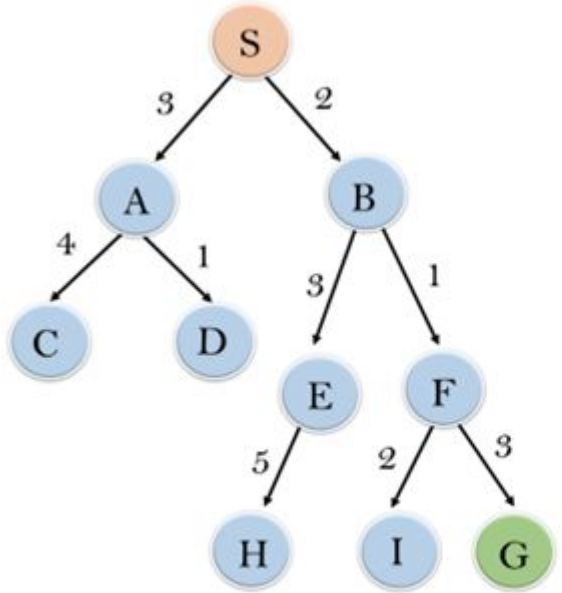
- Greedy best-first search algorithm always selects the path which appears **best at that moment.**
- It uses the heuristic function and search. Best-first search allows us to take the advantages of depth-first search and breadth-first search.
- In the best first search algorithm, we expand the node and the closest cost is estimated by heuristic function, i.e.

$$f(n)= g(n).$$

- Where h(n)= estimated cost from node n to the goal.
- The greedy best-first algorithm is implemented by the priority queue.

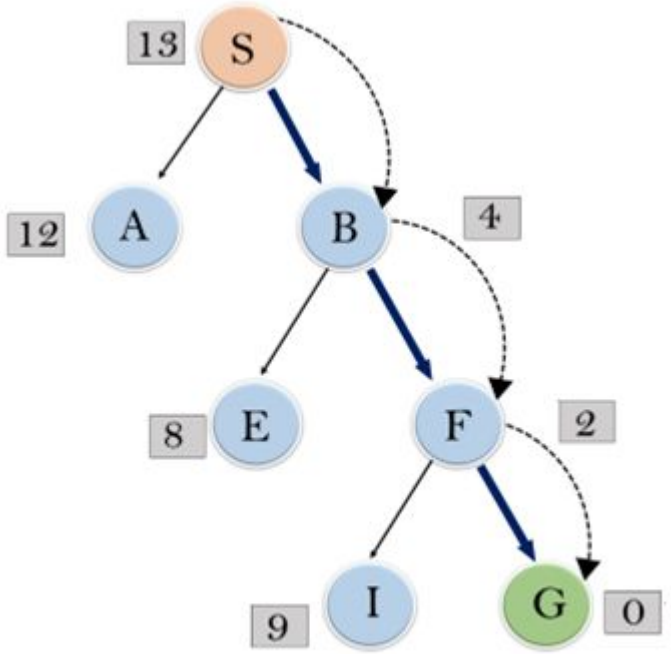
Example

- Consider the below search problem, and we will traverse it using greedy best-first search. At each iteration, each node is expanded using the heuristic function f(n)=h(n) , which is given in the below table.



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

We are using two lists which are OPEN and CLOSED Lists. Following are the iteration for traversing.



Expand the nodes of S and put in the CLOSED list

Initialization: Open [A, B], Closed [S]

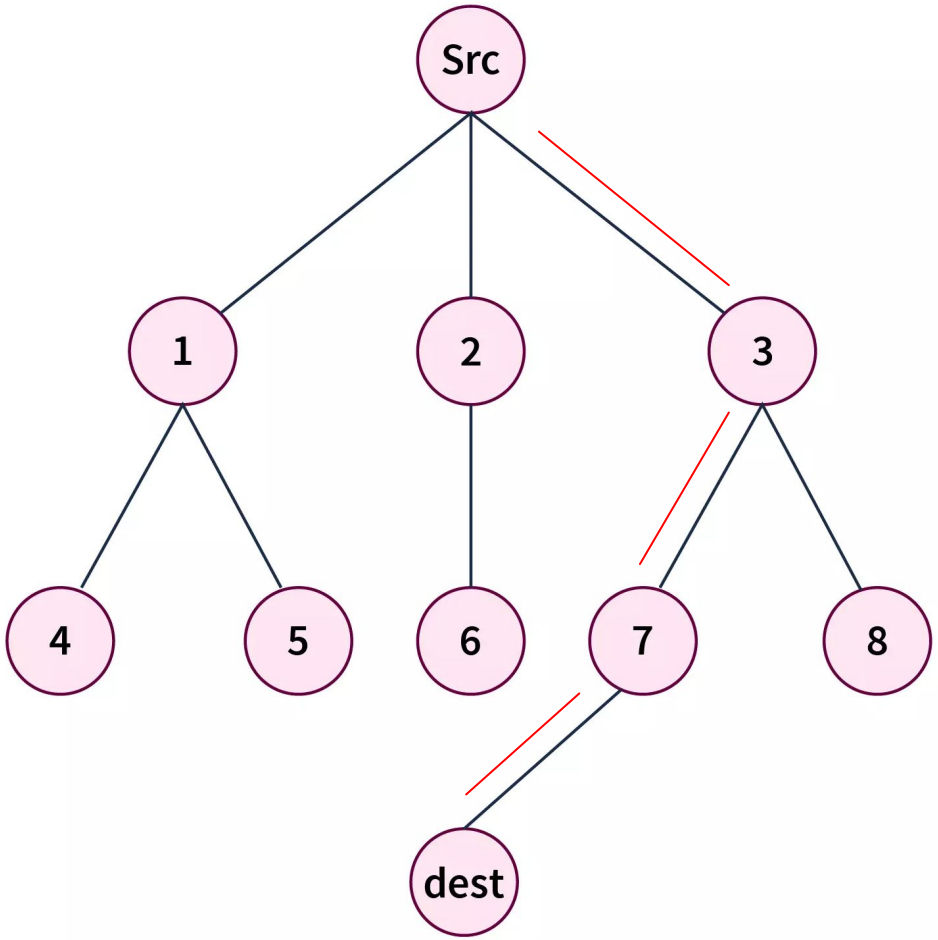
Iteration 1 : Open [A], Closed [S, B]

Iteration 2 : Open [E, F, A], Closed [S, B]
: Open [E, A], Closed [S, B, F]

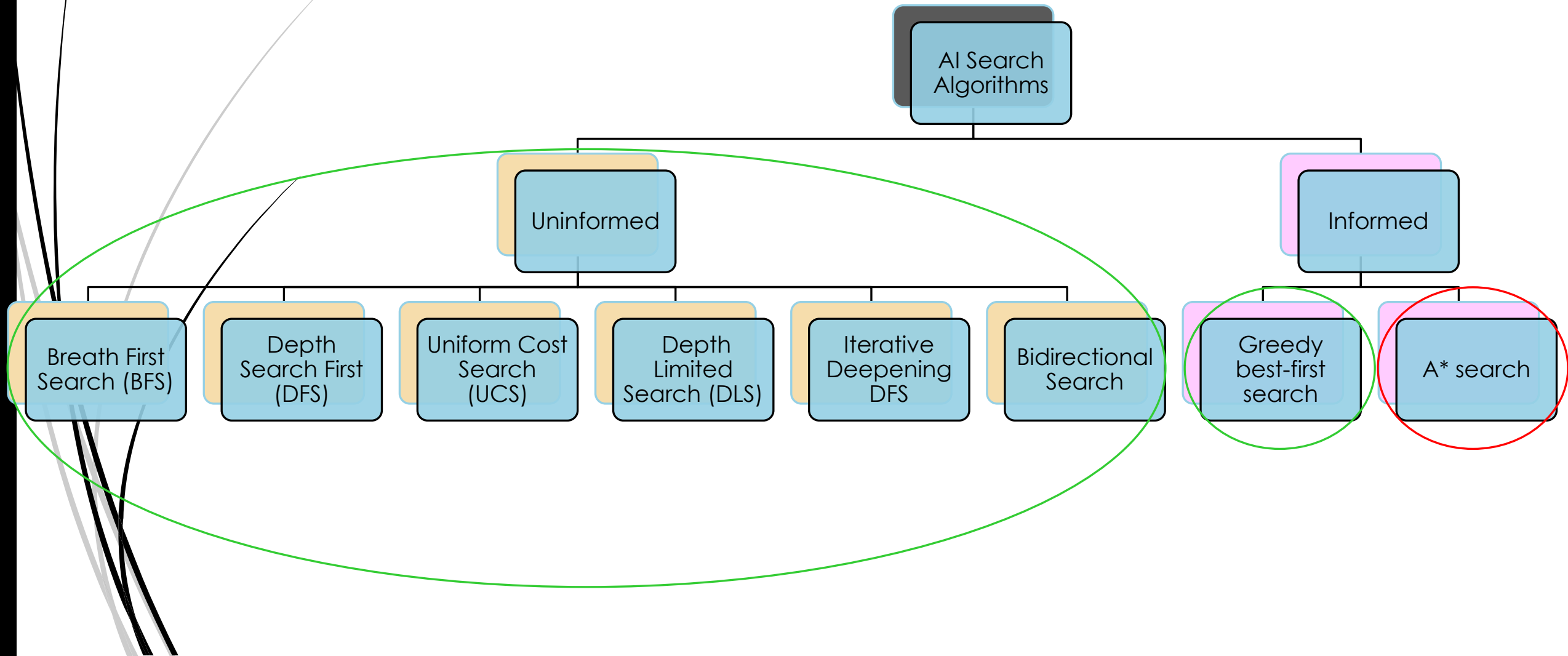
Iteration 3 : Open [I, G, E, A], Closed [S, B, F]
: Open [I, E, A], Closed [S, B, F, G]

Hence the final solution path will be: **S----> B----->F----> G**

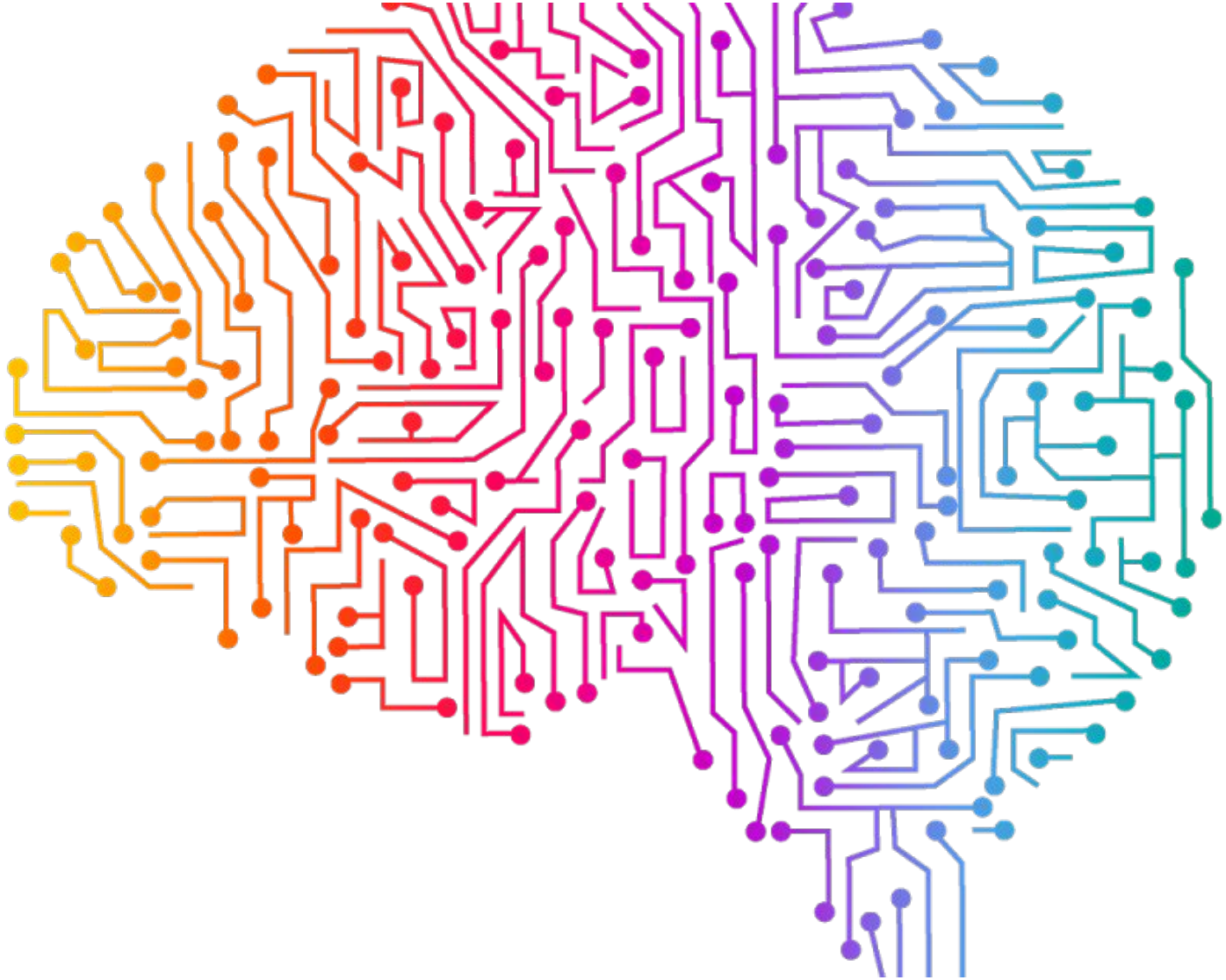
Activity



Src	20
1	22
2	21
3	10
4	25
5	24
6	30
7	5
8	12
dest	0


AI Search Algorithms
Uninformed
Informed
Breath First Search (BFS)
Depth Search First (DFS)
Uniform Cost Search (UCS)
Depth Limited Search (DLS)
Iterative Deepening DFS
Bidirectional Search
Greedy best-first search
A* search

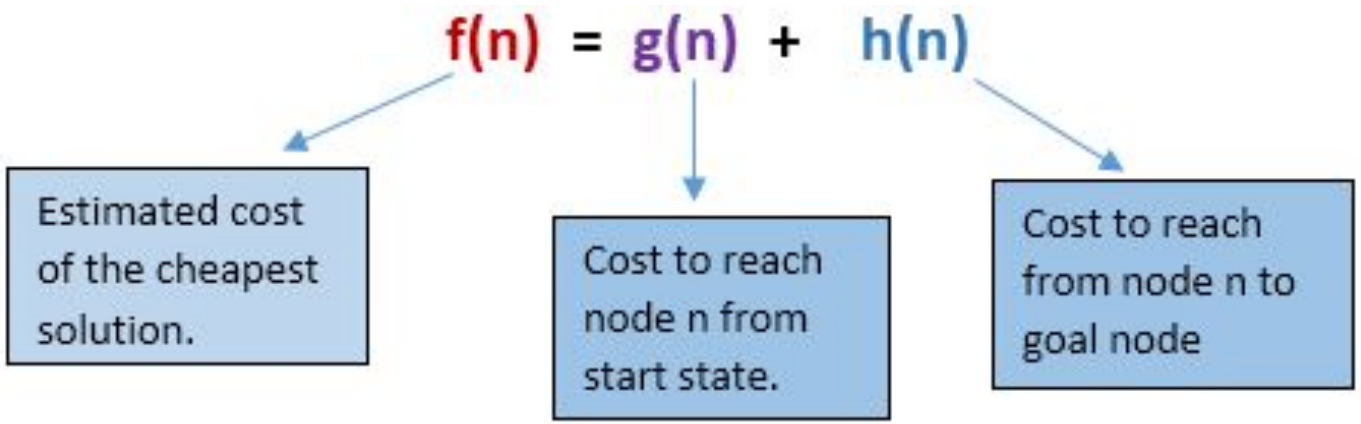
A* Search



A* Search

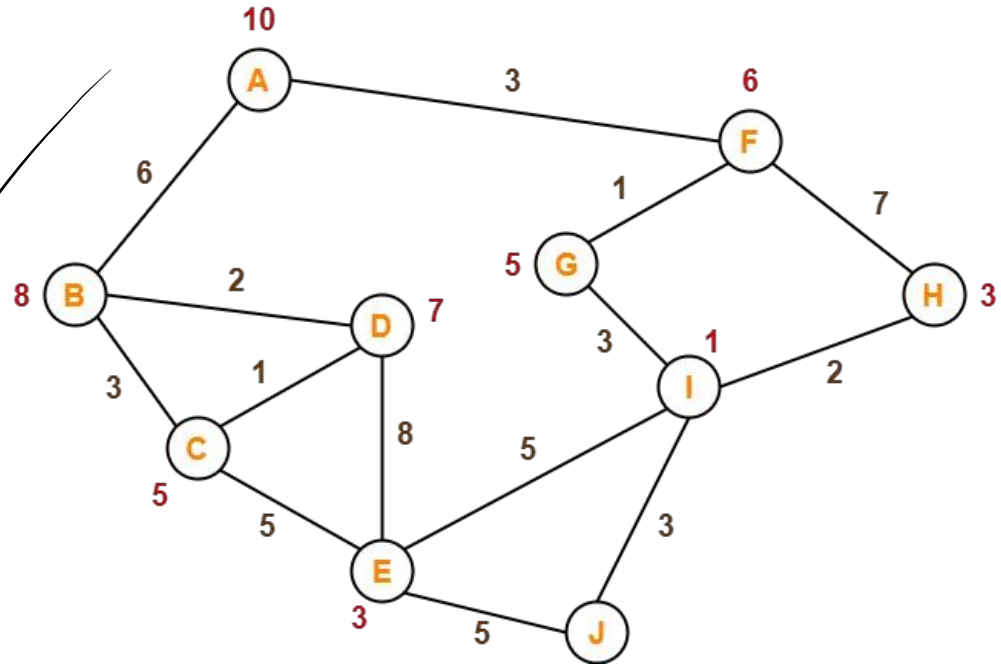
- A* search is the most commonly known form of best-first search.
- It uses heuristic function h(n), and cost to reach the node n from the start state g(n).
- It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently.
- A* search algorithm finds the shortest path through the search space using the heuristic function.
- This search algorithm expands less search tree and provides optimal result faster.

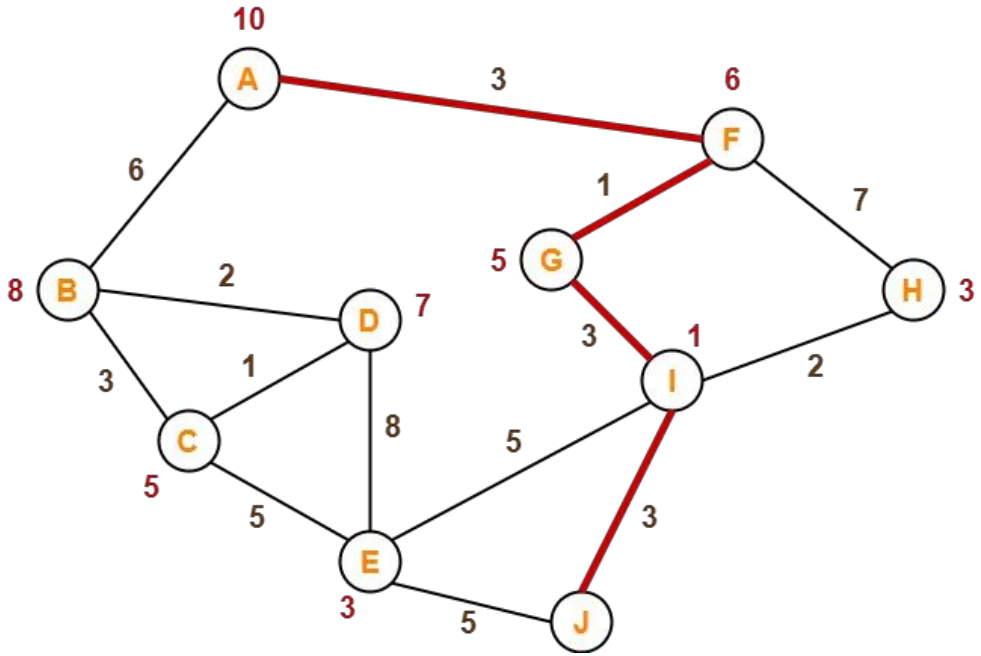
- In A* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called as a fitness number.



Activity

(The value on the nodes is h(n) and value on the edges is g(n))





Step-01:

- We start with node A.
- Node B and Node F can be reached from node A.
- A* Algorithm calculates

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

- $f(B) = g(B) + h(B) = 6 + 8 = 14$
- $f(F) = g(F) + h(F) = 3 + 6 = 9$
- Since $f(F) < f(B)$, so it decides to go to node F.
- Path- A → F

Step-02:

- Node G and Node H can be reached from node F.
- A* Algorithm calculates

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

- $f(G) = g(G) + h(G) = (3+1) + 5 = 9$
- $f(H) = g(H) + h(H) = (3+7) + 3 = 13$
- Since $f(G) < f(H)$, so it decides to go to node G.
- Path - A → F → G

Step-03:

- Node I can be reached from node G.
- A* Algorithm calculates

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

- $f(I) = g(I) + h(I)$

$$= (3+1+3) + 1 = 8$$

- So it decides to go to node I.
- Path - A → F → G → I

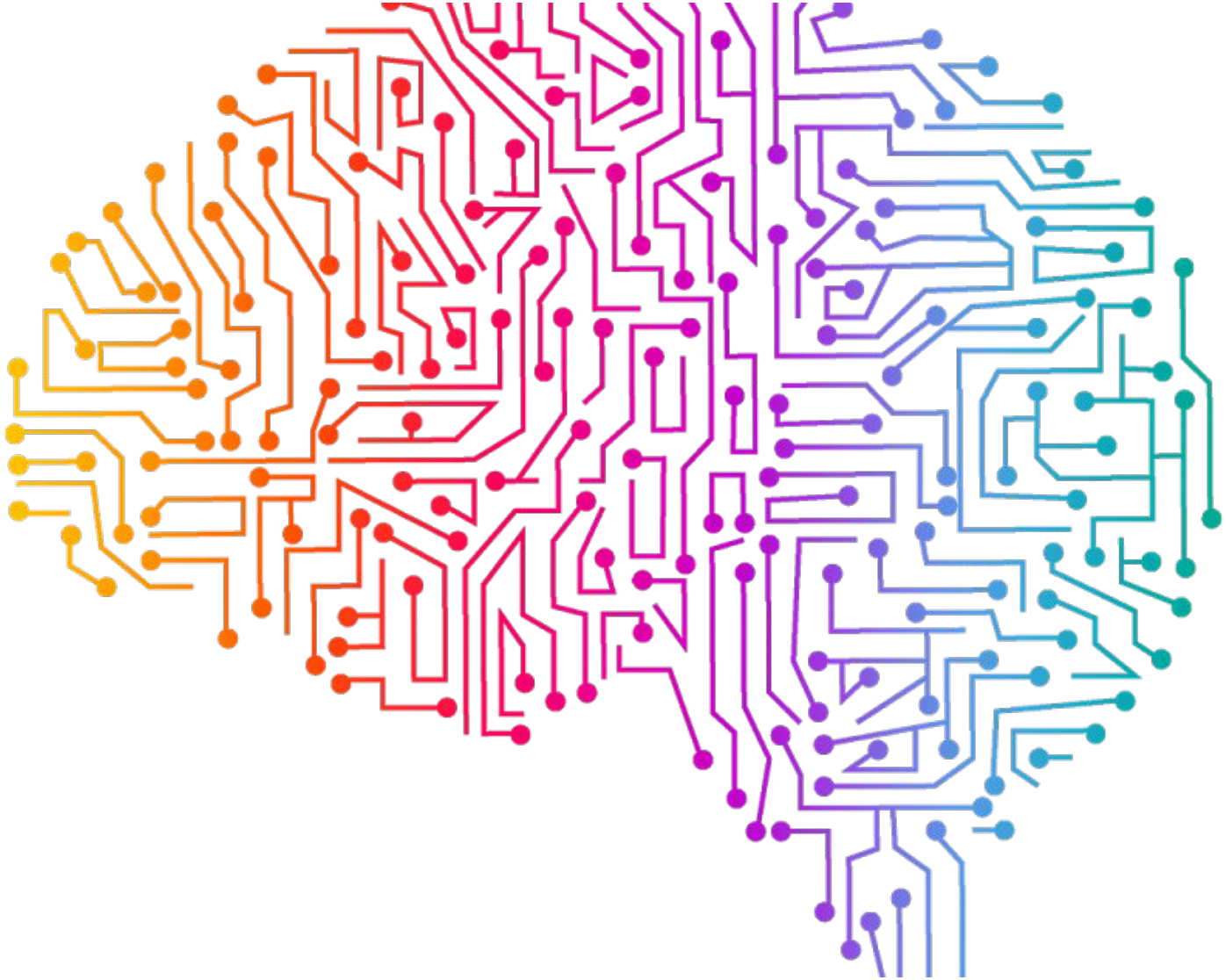
Step-04:

- Node E, Node H and Node J can be reached from node I.
- A* Algorithm calculates

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

- $f(E) = (3+1+3+5) + 3 = 15$
- $f(H) = (3+1+3+2) + 3 = 12$
- $f(J) = (3+1+3+3) + 0 = 10$
- Since f(J) is least, so it decides to go to node J.
- Path - A → F → G → I → J

Advantages and Disadvantages of Informed Search



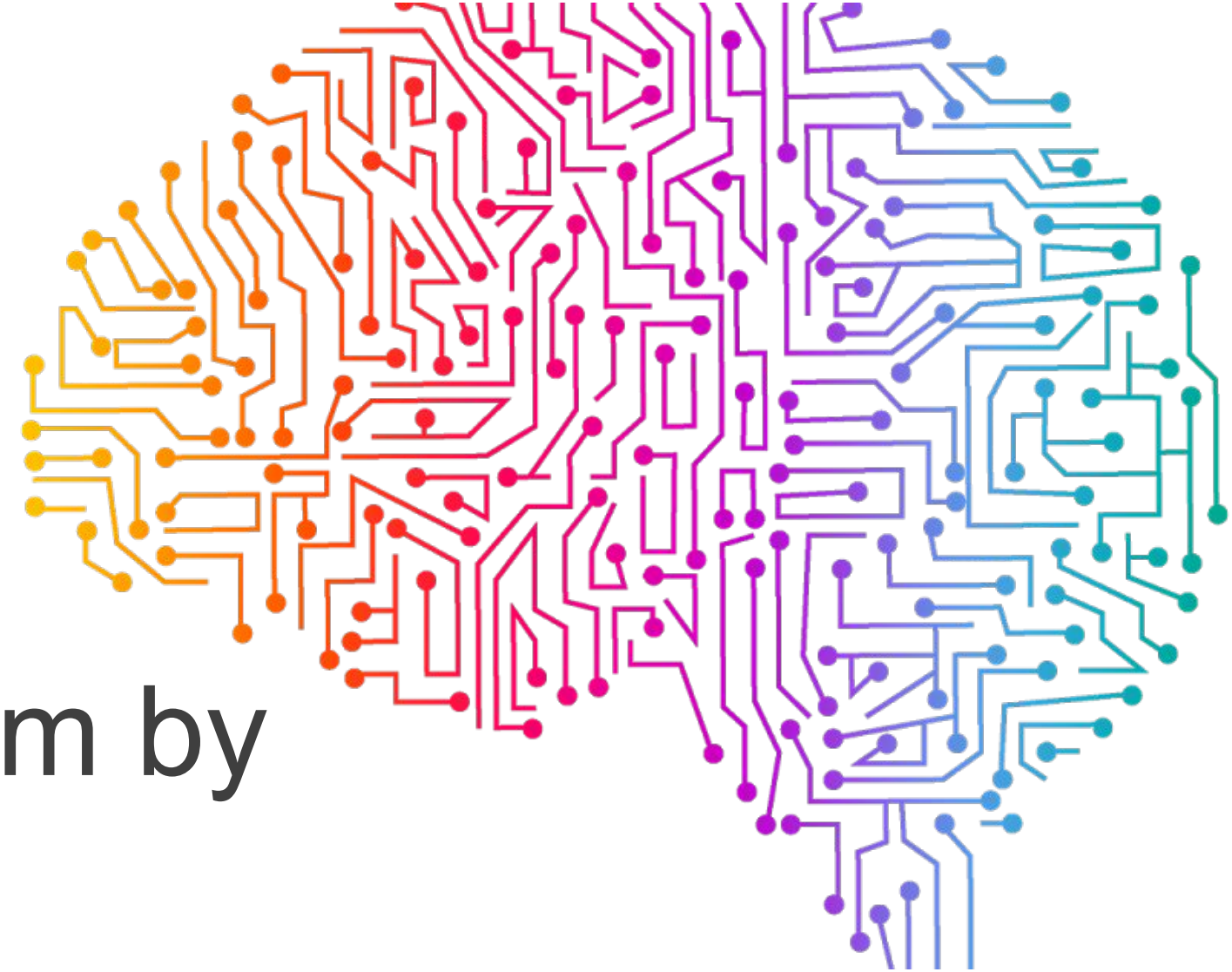
Advantages and Disadvantages

Advantages:

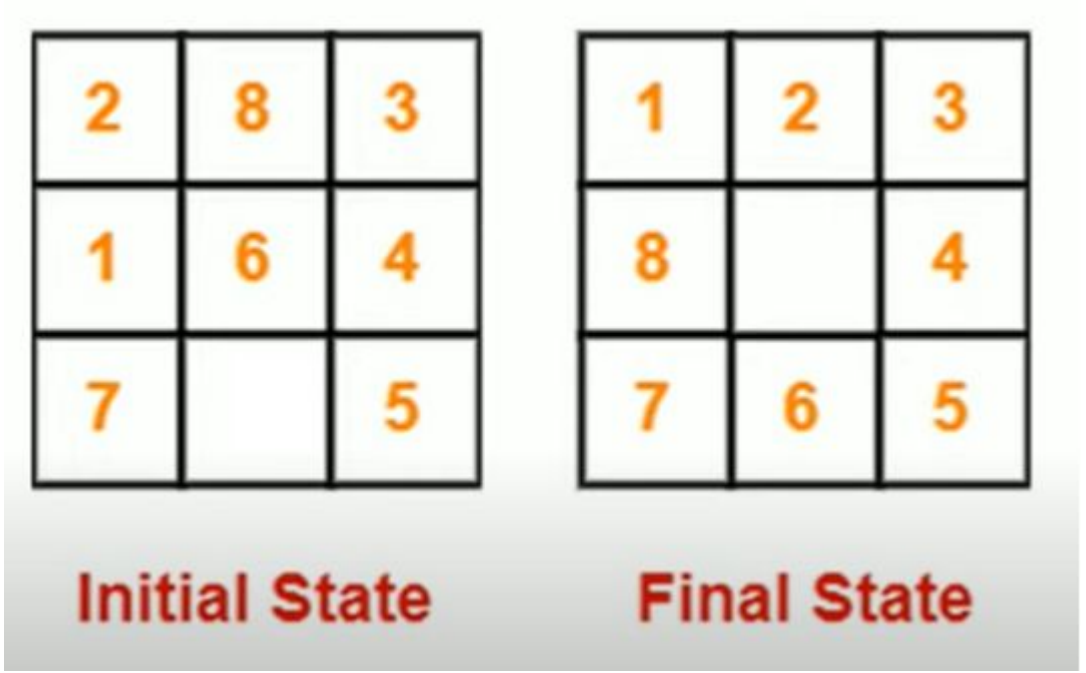
- Informed search can switch between BFS and DFS by gaining the advantages of both algorithms.
- Informed Search is more efficient than Uninformed Search.
- A* search algorithm is the best algorithm than other search algorithms.
- A* search algorithm is optimal and complete.
- A* search algorithm can solve very complex problems.

Disadvantages:

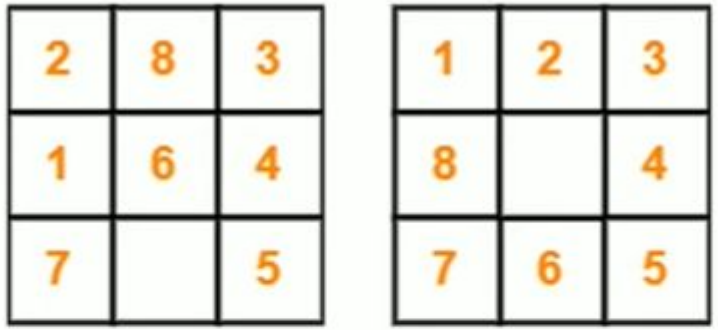
- Greedy Best First is not optimal.
- A* search does not always produce the shortest path as it mostly based on heuristics and approximation.
- The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.



Puzzle Problem by A* Search


2 8 3
1 6 4
7 5
1 2 3
8 4
7 6 5
Initial State
Final State

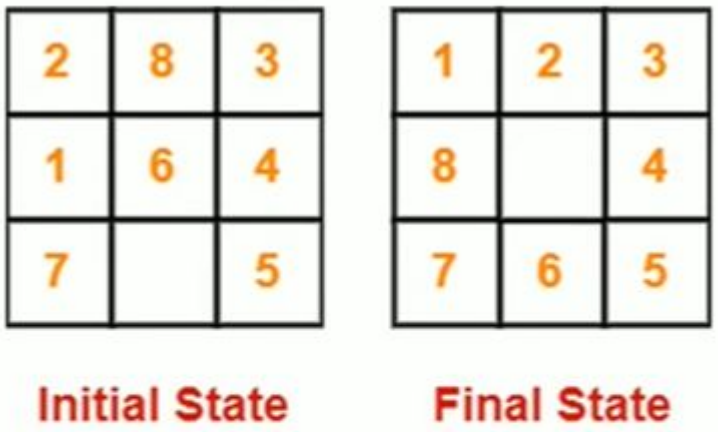
Given an initial state of a 8-puzzle problem and final state to be reached-

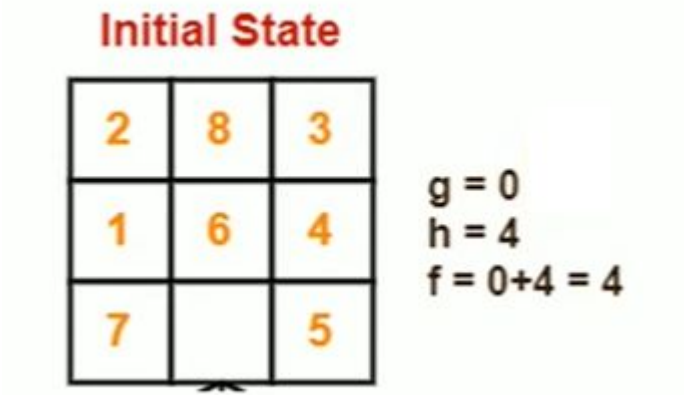


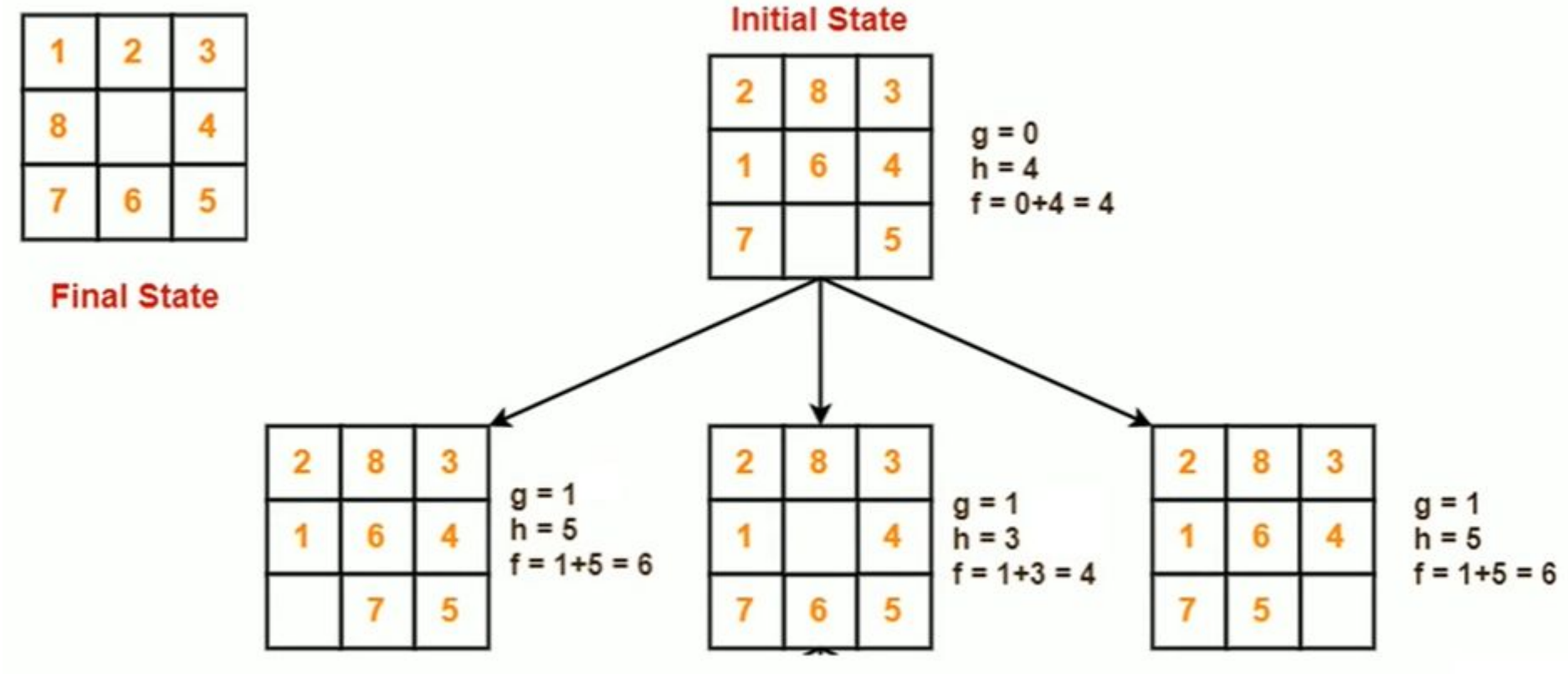
- Find the most cost-effective path to reach the final state from initial state using A* Algorithm.

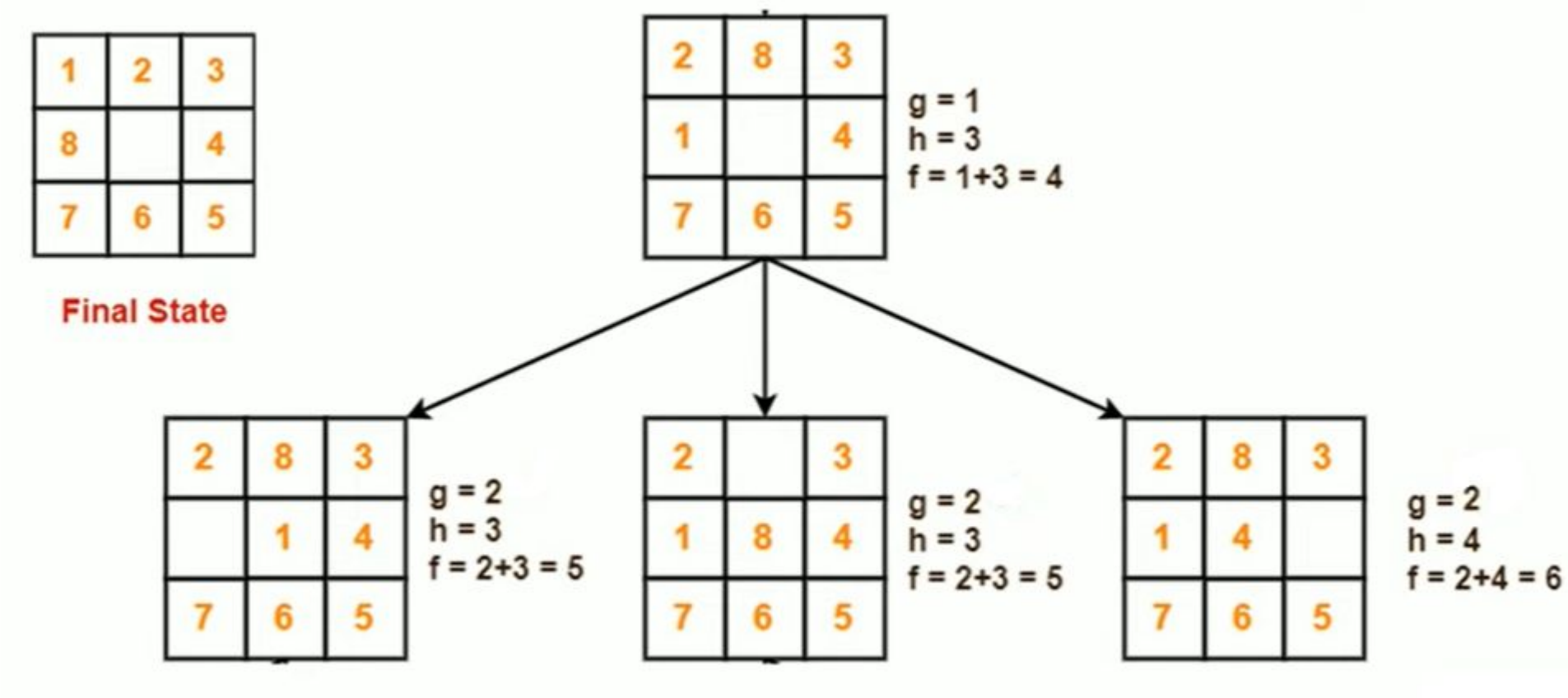
$$f(n) = g(n) + h(n)$$

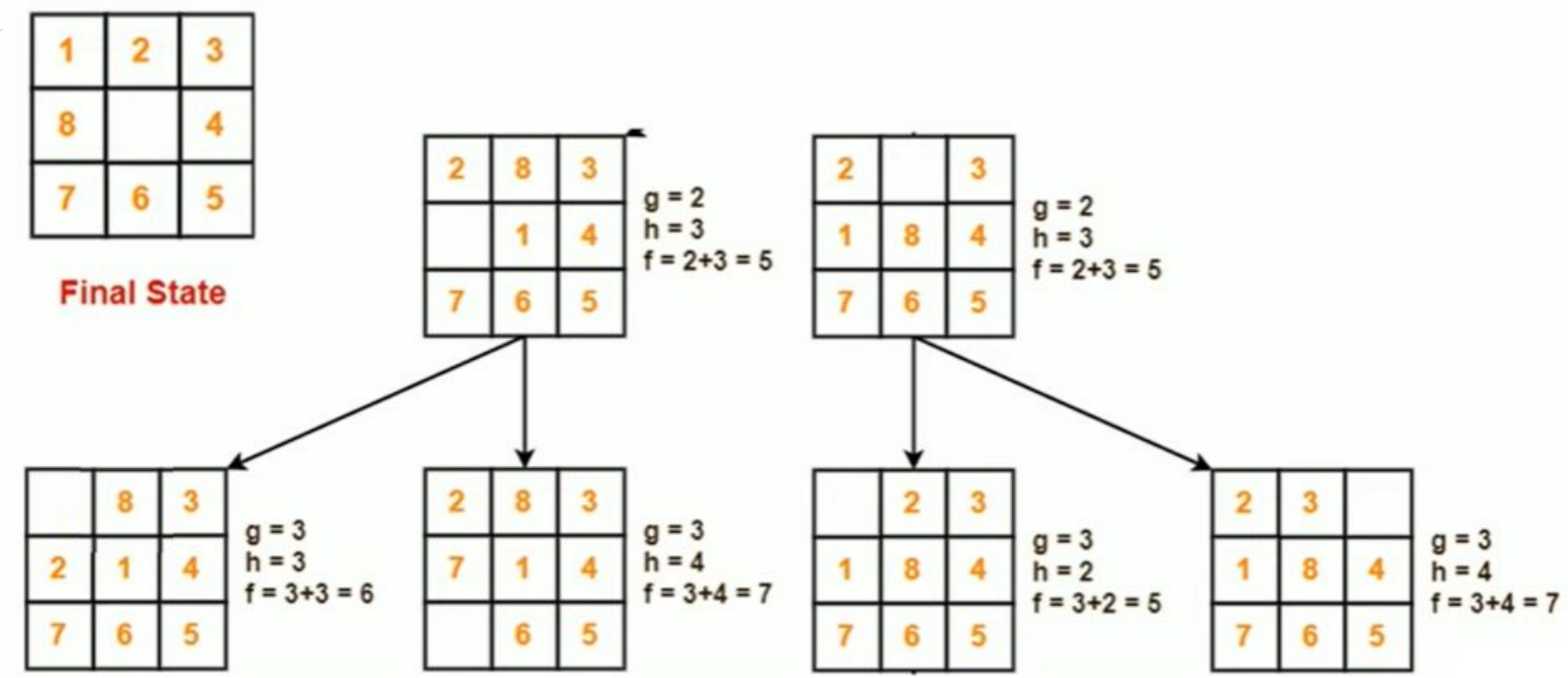
- Consider **g(n) = Depth of node** and **h(n) = Number of misplaced tiles**.


2 8 3
1 6 4
7 5
1 2 3
8 4
7 6 5
Initial State
Final State
g = 0
h = 4

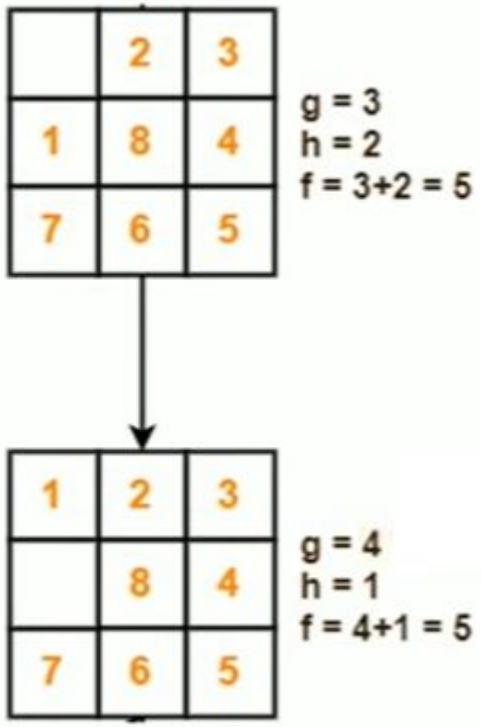

Initial State
2 8 3
1 6 4
7 5
g = 0
h = 4
f = 0+4 = 4

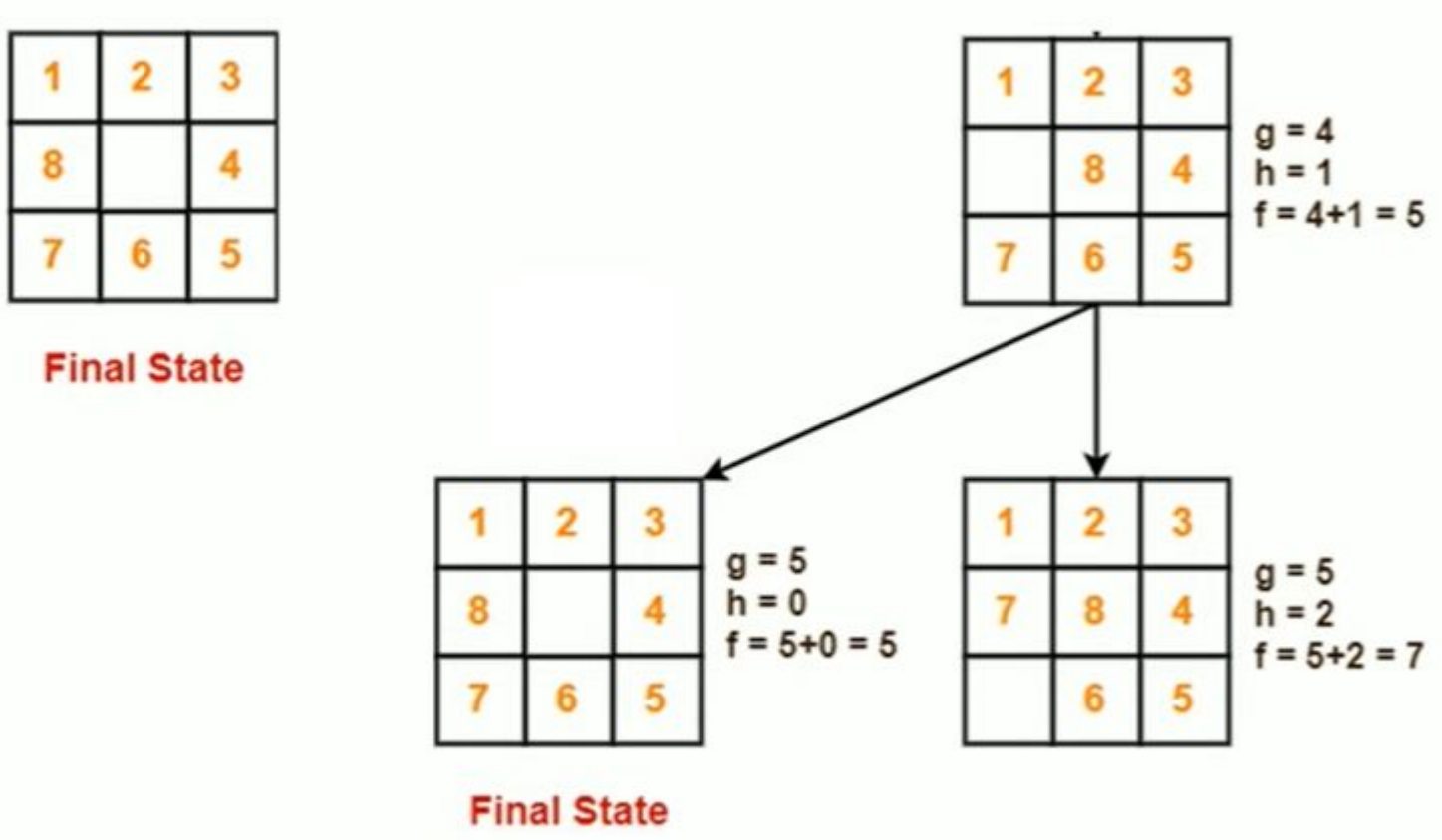

1 2 3
8 4
7 6 5
Final State
Initial State
2 8 3
1 6 4
7 5
g = 0
h = 4
f = 0+4 = 4
2 8 3
1 6 4
7 5
g = 1
h = 5
f = 1+5 = 6
2 8 3
1 4
7 6 5
g = 1
h = 3
f = 1+3 = 4
2 8 3
1 6 4
7 5
g = 1
h = 5
f = 1+5 = 6

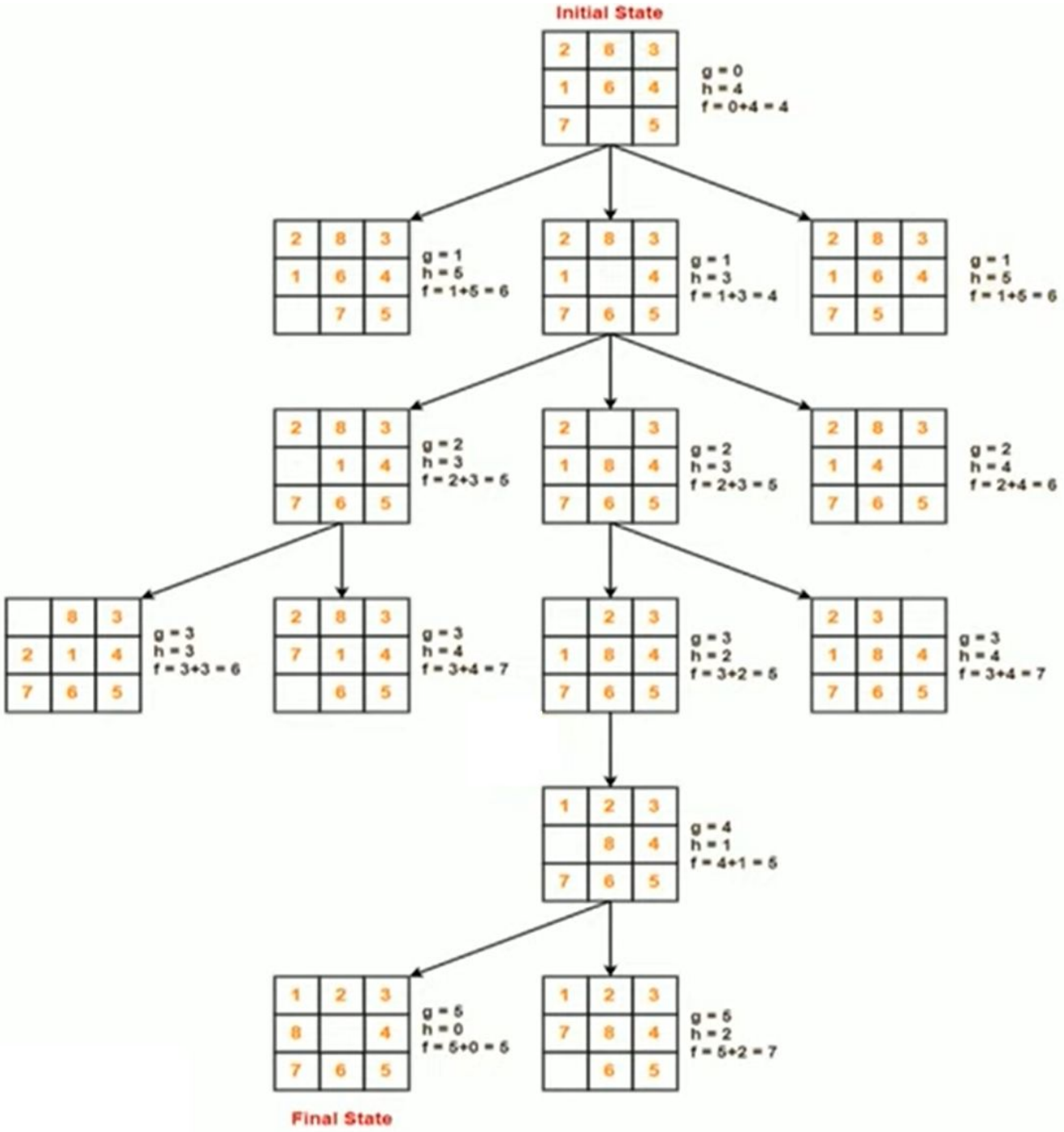

1 2 3
8 4
7 6 5
Final State
2 8 3
1 4
7 6 5
g = 1
h = 3
f = 1+3 = 4
2 8 3
1 4
7 6 5
g = 2
h = 3
f = 2+3 = 5
2 3
1 8 4
7 6 5
g = 2
h = 3
f = 2+3 = 5
2 8 3
1 4
7 6 5
g = 2
h = 4
f = 2+4 = 6


1 2 3
8 4
7 6 5
Final State
2 8 3
1 4
7 6 5
g = 2
h = 3
f = 2+3 = 5
2 3
1 8 4
7 6 5
g = 2
h = 3
f = 2+3 = 5
8 3
2 1 4
7 6 5
g = 3
h = 3
f = 3+3 = 6
2 8 3
7 1 4
6 5
g = 3
h = 4
f = 3+4 = 7
2 3
1 8 4
7 6 5
g = 3
h = 2
f = 3+2 = 5
2 3
1 8 4
7 6 5
g = 3
h = 4
f = 3+4 = 7

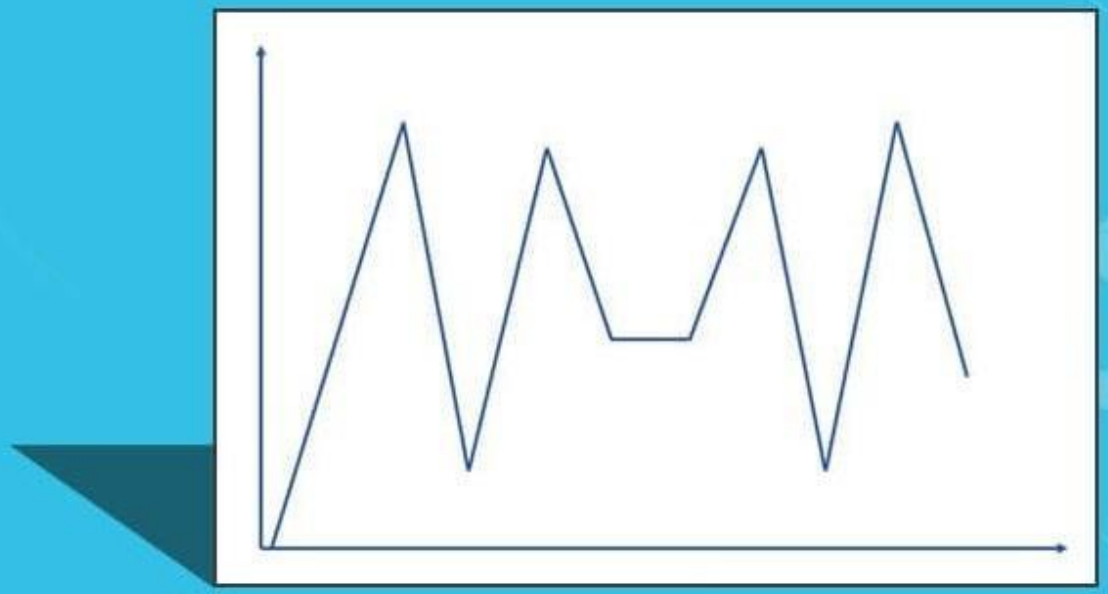
Final State




1 2 3
8 4
7 6 5
Final State
1 2 3
8 4
7 6 5
g = 4
h = 1
f = 4+1 = 5
1 2 3
8 4
7 6 5
g = 5
h = 0
f = 5+0 = 5
1 2 3
7 8 4
6 5
g = 5
h = 2
f = 5+2 = 7
Final State


Initial State
g = 0
h = 4
f = 0+4 = 4
g = 1
h = 5
f = 1+5 = 6
g = 1
h = 3
f = 1+3 = 4
g = 1
h = 5
f = 1+5 = 6
g = 2
h = 3
f = 2+3 = 5
g = 2
h = 3
f = 2+3 = 5
g = 2
h = 4
f = 2+4 = 6
g = 3
h = 3
f = 3+3 = 6
g = 3
h = 4
f = 3+4 = 7
g = 3
h = 2
f = 3+2 = 5
g = 3
h = 4
f = 3+4 = 7
g = 4
h = 1
f = 4+1 = 5
g = 5
h = 0
f = 5+0 = 5
g = 5
h = 2
f = 5+2 = 7
Final State

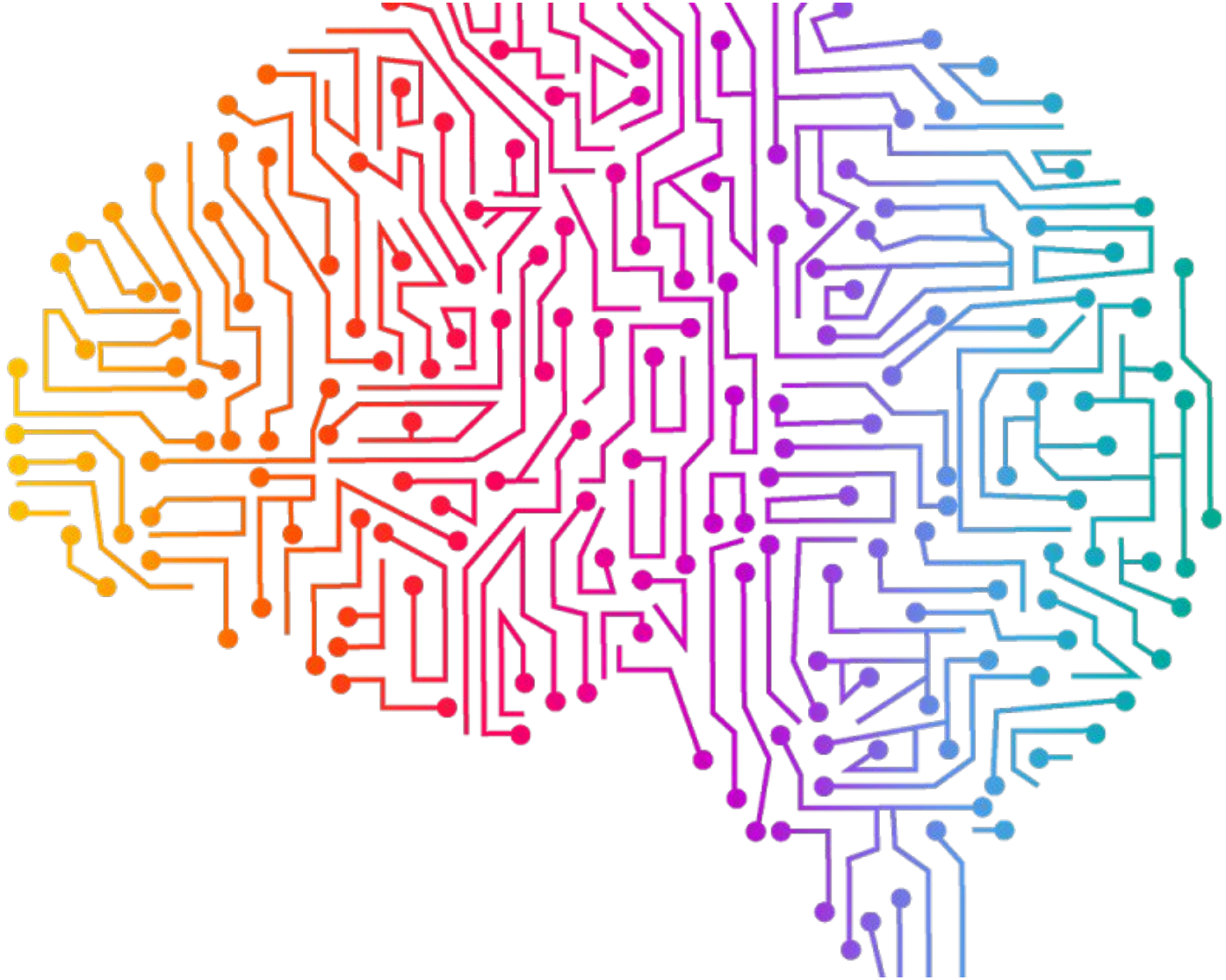
LOCAL SEARCH (HILL CLIMBING ALGORITHM)



Learning Objective of this Topic

- What is Local Search?
- Hill Climbing Algorithm
- Hill Climbing Search Example using Local and Global Heuristic Function
- Advantages and Disadvantages of Hill Climbing (with Real-Life Applications)

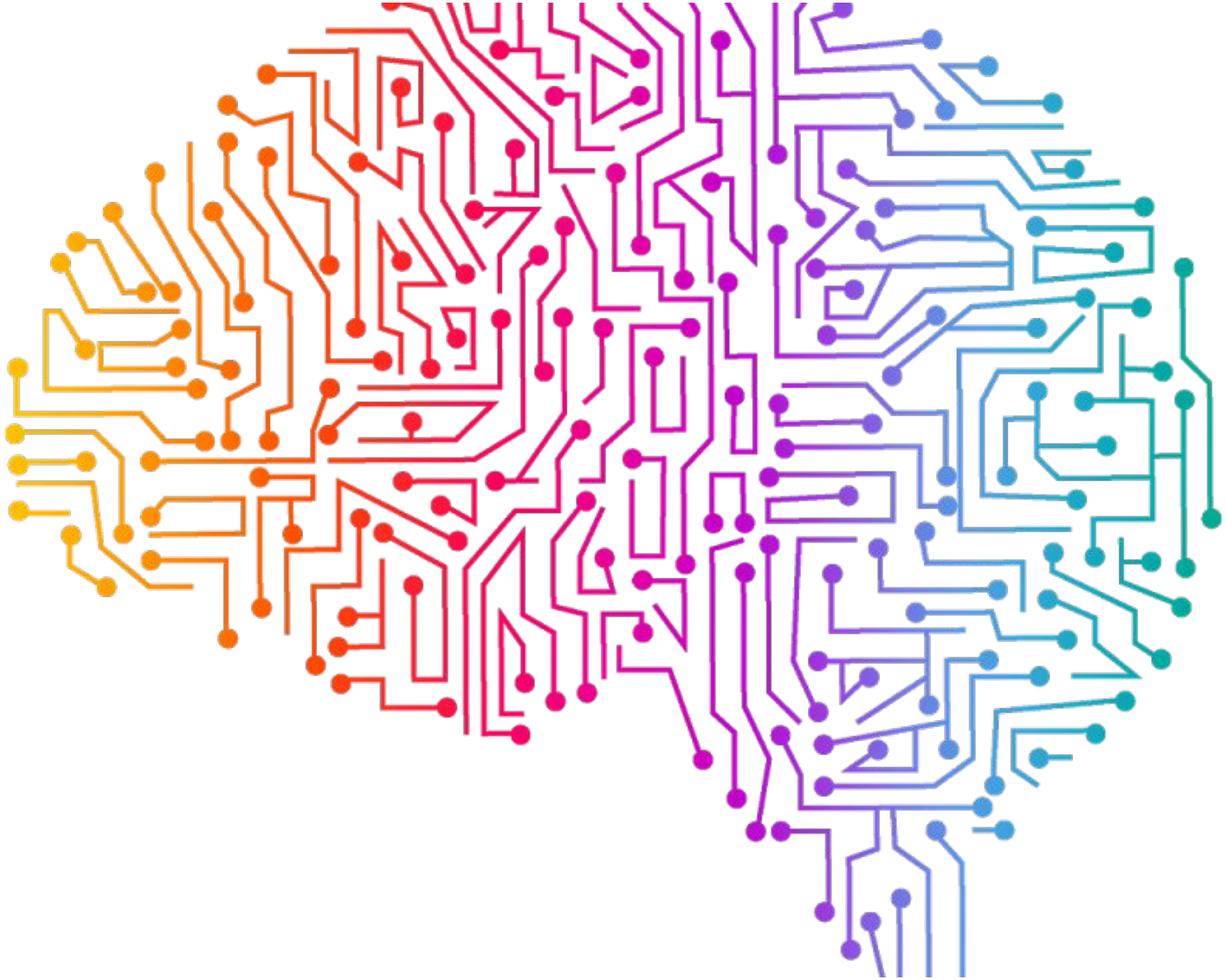
What is Local Search?



Local Search

- Local search algorithm focus on finding solutions:
  - within a limited part of the solution space,
  - making incremental improvements to a current solution
  - until reaching a satisfactory outcome.
  - They don't explore the entire solution space.

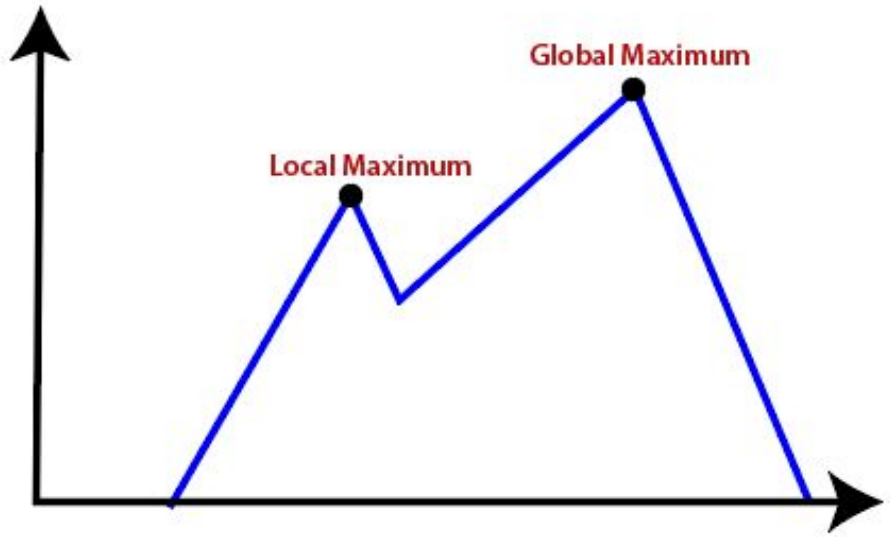
Hill Climbing Algorithm



Hill Climbing

- A hill-climbing algorithm is a local search algorithm that moves continuously upward (increasing) until the best solution is attained. This algorithm comes to an end when the peak is reached.
- It begins with a non-optimal state (the hill's base) and upgrades this state until a certain precondition is met.
- The heuristic function is used as the basis for this precondition.
- The continuous improvement of the current state of iteration can be termed climbing.
- This explains why the algorithm is termed a *hill-climbing algorithm*.

- **Local Maximum:** Local maximum is a state which is better than its neighbor states, but there is also another state which is higher than it.
- **Global Maximum:** Global maximum is the best possible state of state space landscape. It has the highest value of an objective function. (Goal State)
- **Current state:** It is a state where an agent is currently present.

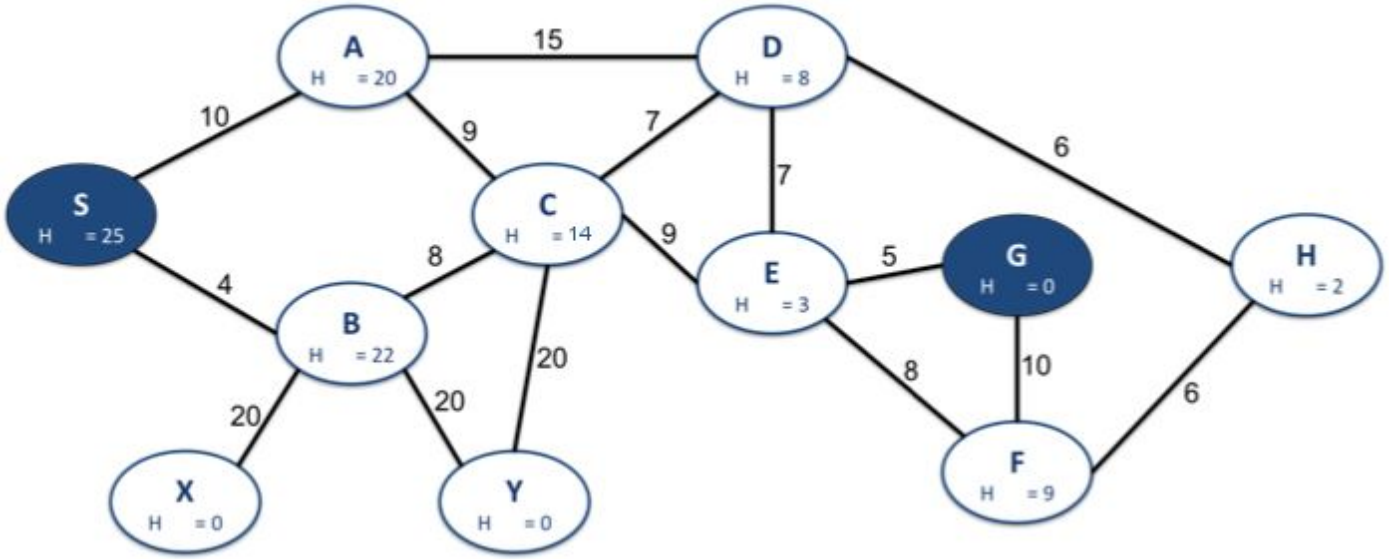


Features of Hill Climbing

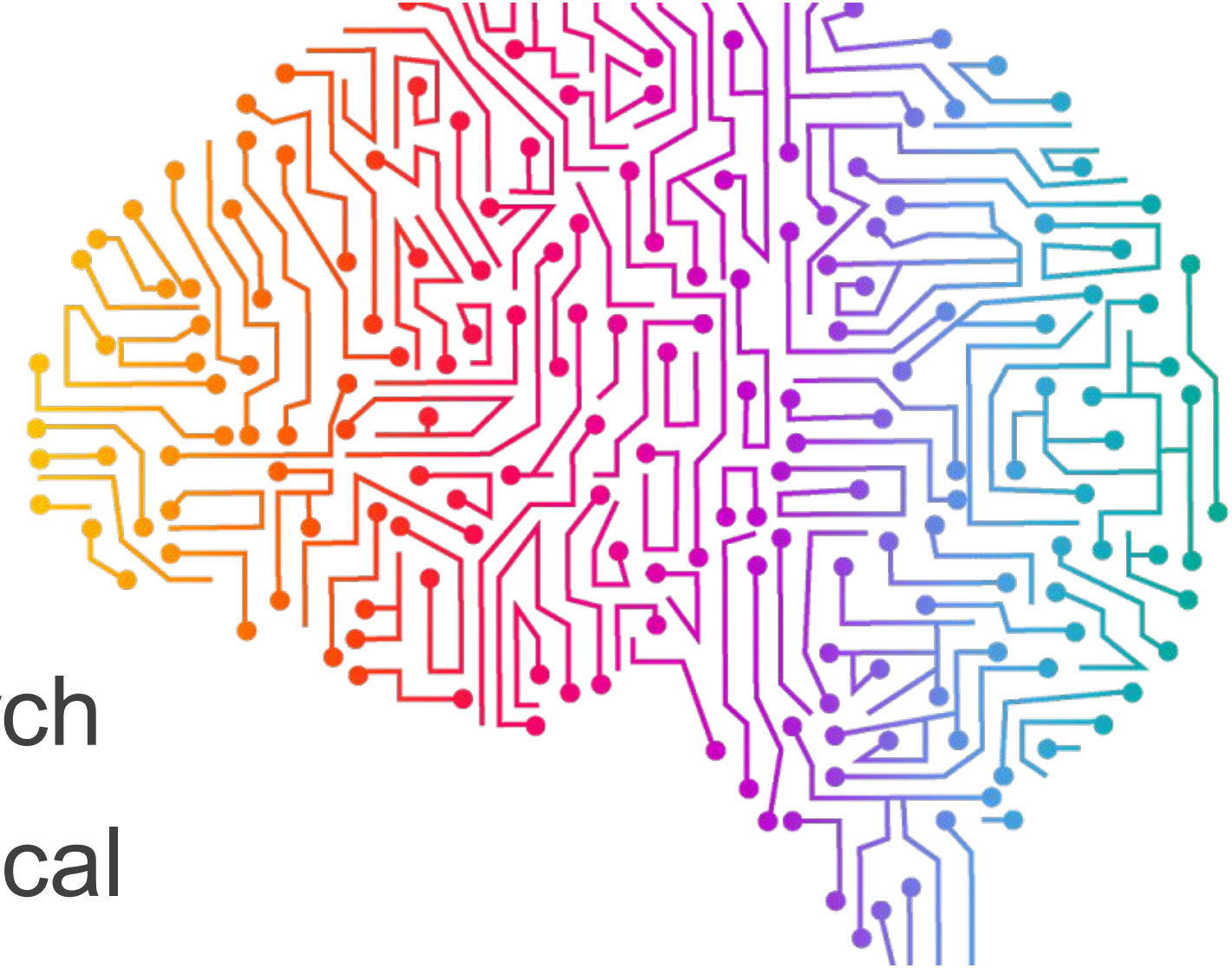
A hill-climbing algorithm has some main features:

- It employs a **greedy approach:** This means that it moves in a direction in which the cost function is optimized. The greedy approach enables the algorithm to establish local maxima or minima.
- **No Backtracking:** A hill-climbing algorithm only works on the current state and succeeding states (future). It does not look at the previous states.
- **Feedback mechanism:** The algorithm has a feedback mechanism that helps it decide on the direction of movement (whether up or down the hill). The feedback mechanism is enhanced through the generate-and-test technique.
- **Blind Search:** We don't know about the whole solution space or about the goal state.

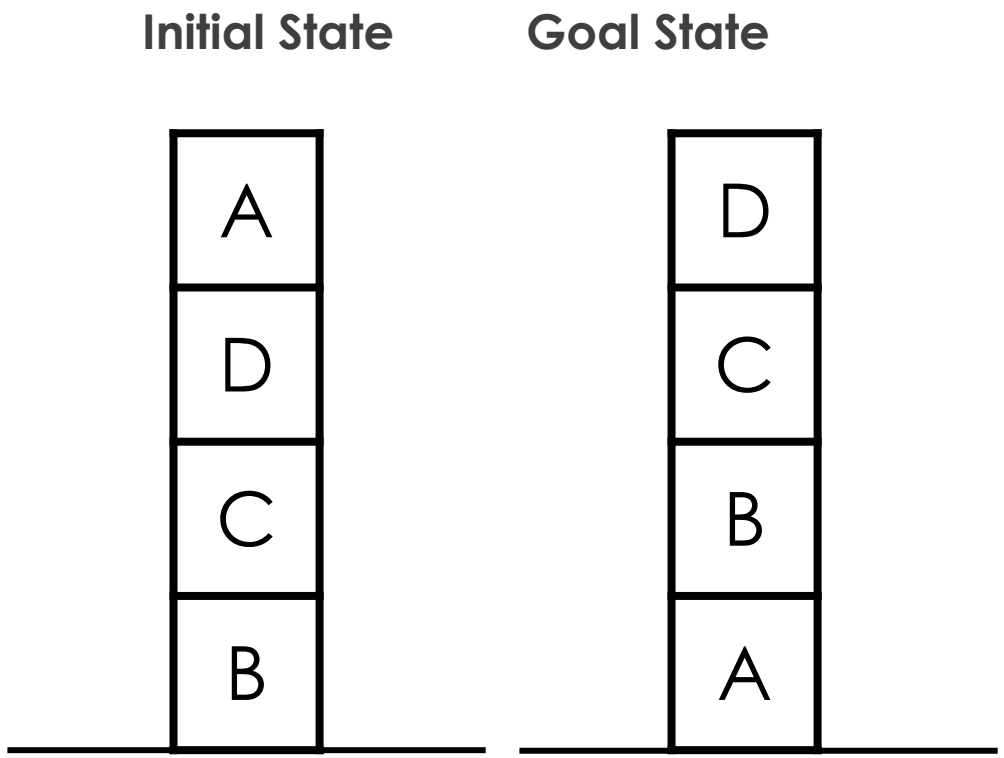
Hill Climbing Search Example



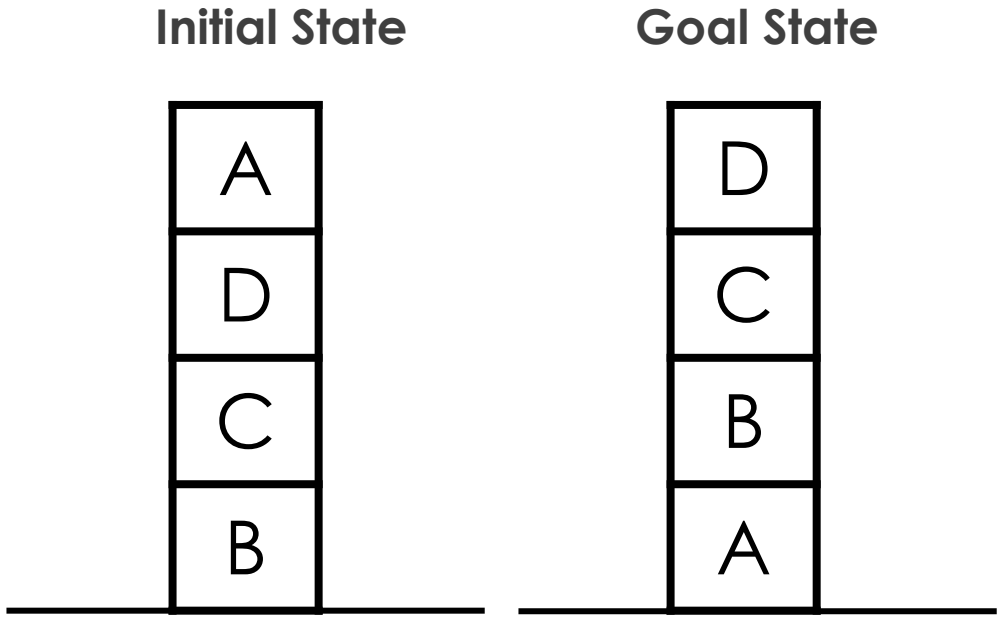
We will go for node with less heuristic value.



Hill Climbing Search Example using Local and Global Heuristic


Initial State
Goal State
A
D
C
B
D
C
B
A

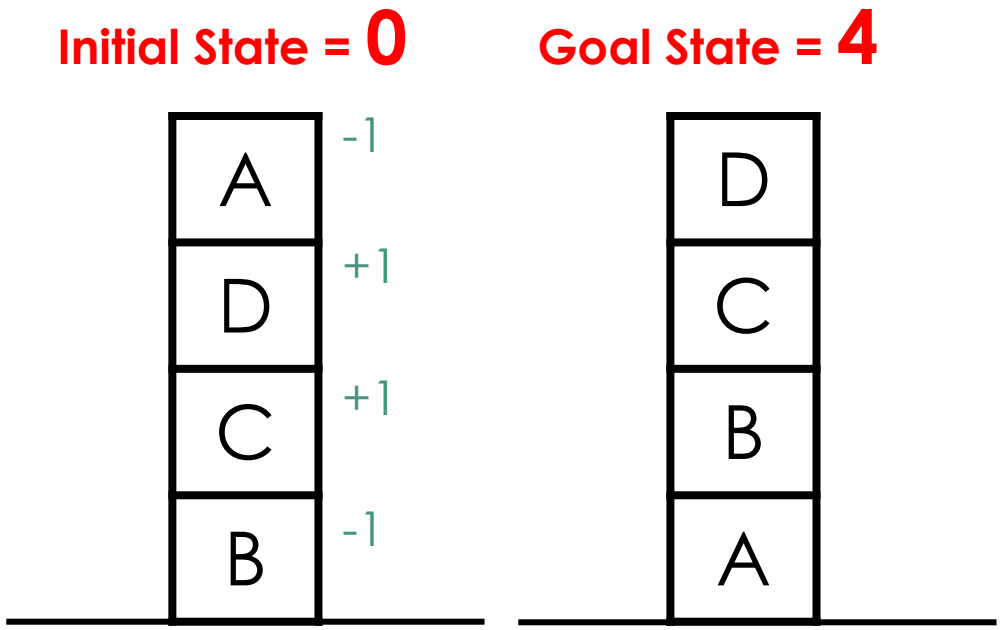
Hill Climbing: Local Heuristic Function



Local Heuristic:

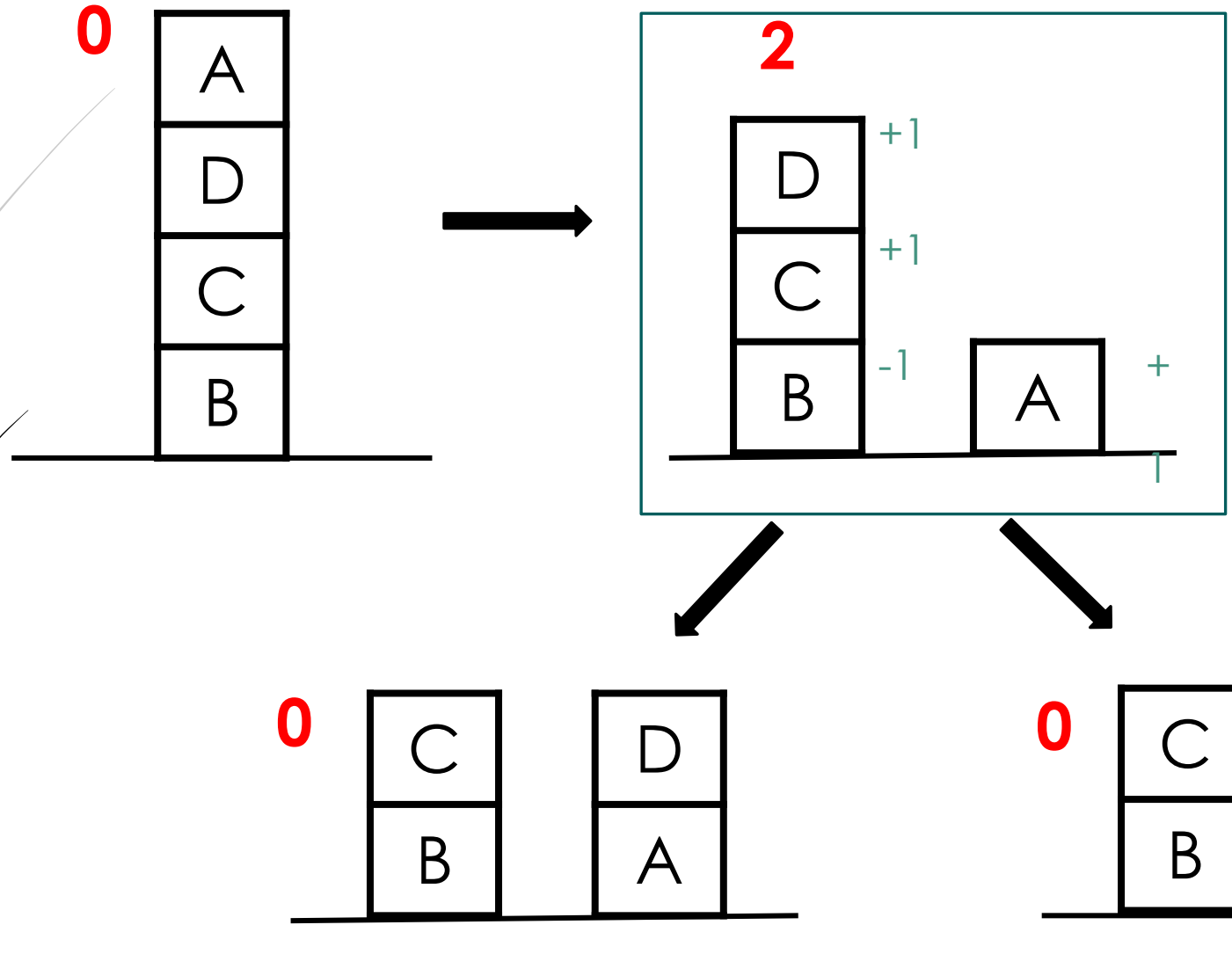
- +1 for each block that is resting on the thing it is supposed to be resting on
- -1 for each block that is resting on the wrong thing
- Operator: Bring any block to the ground or on another block

Hill Climbing: Local Heuristic Function



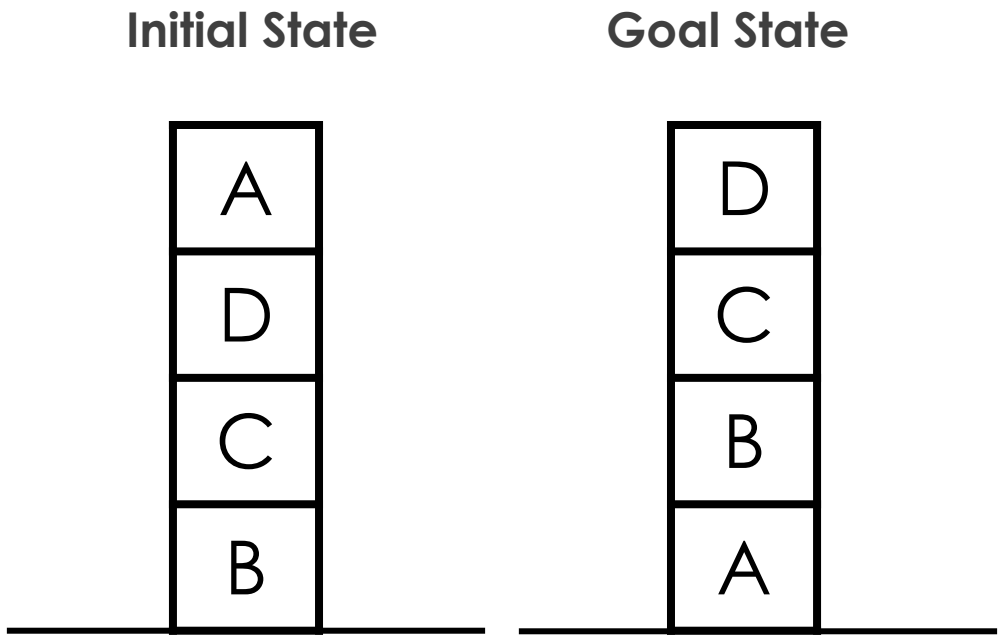
Local Heuristic:

- +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 the wrong thing
- Operator: Bring any block to the ground or on another block



Local Maxima

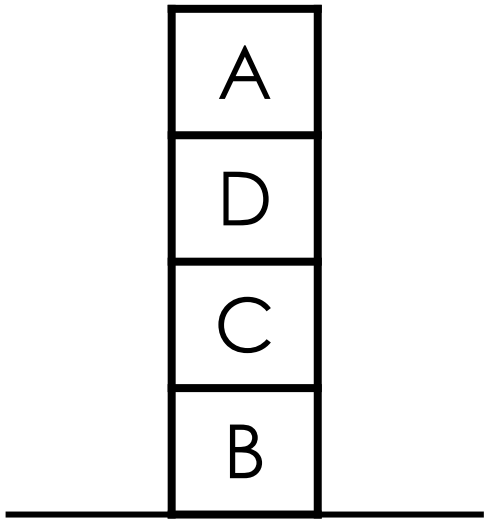
Hill Climbing: Global Heuristic Function



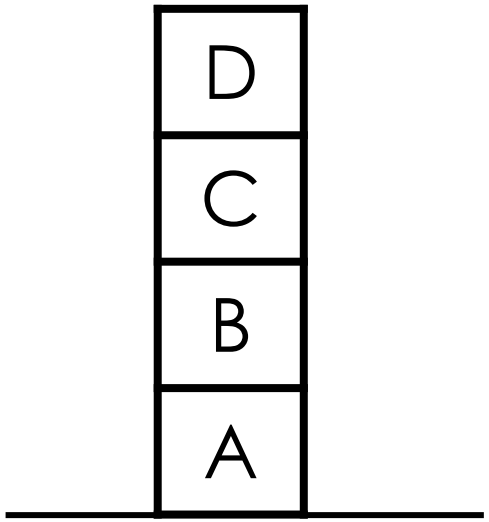
Global Heuristic:

- for each block that has the correct support structure: +1 to every block in the support structure
- for each block that has the wrong support structure: -1 to every block in the support structure
- Operator: Bring any block to the ground or on another block

Initial State = -6

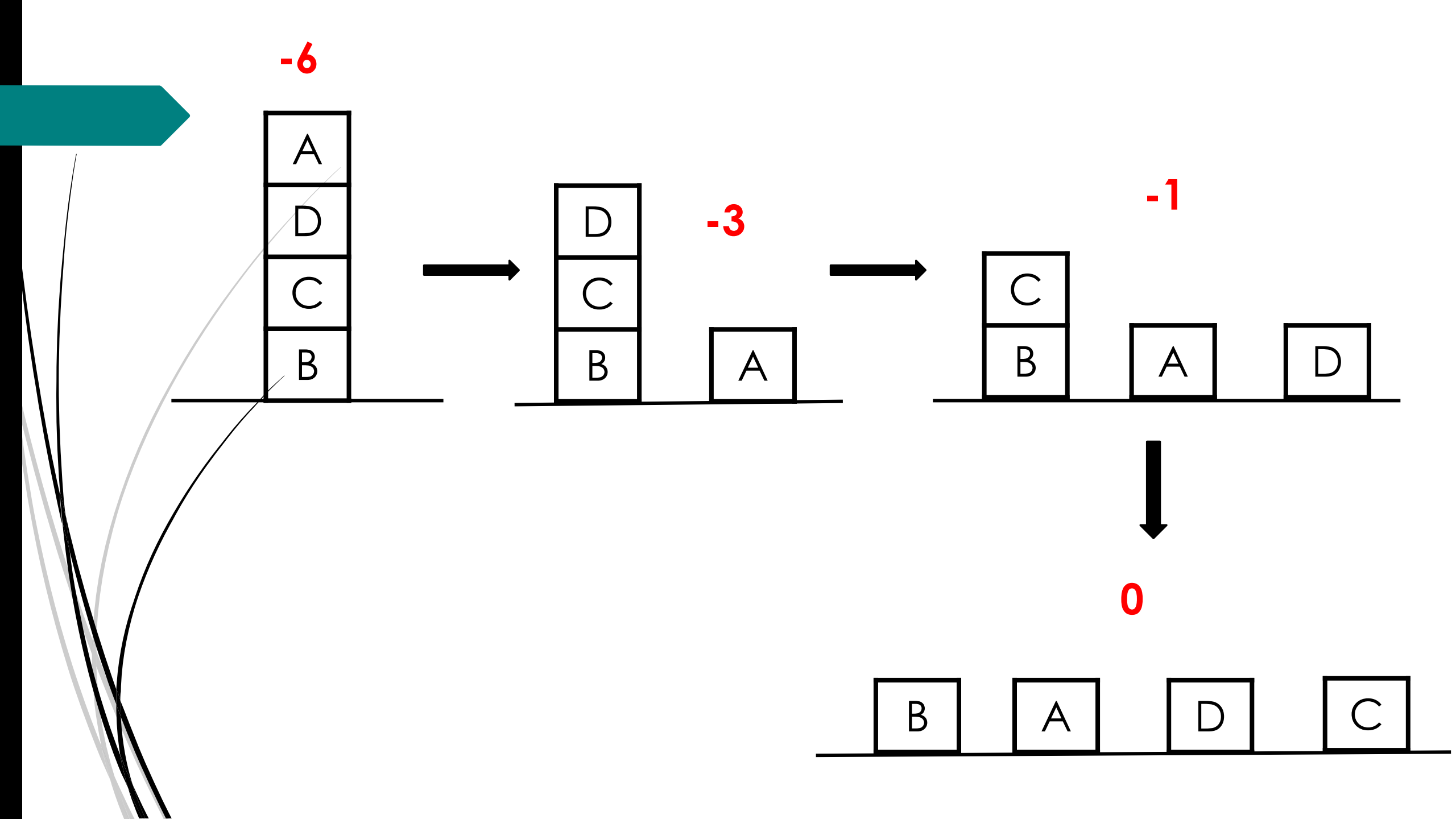


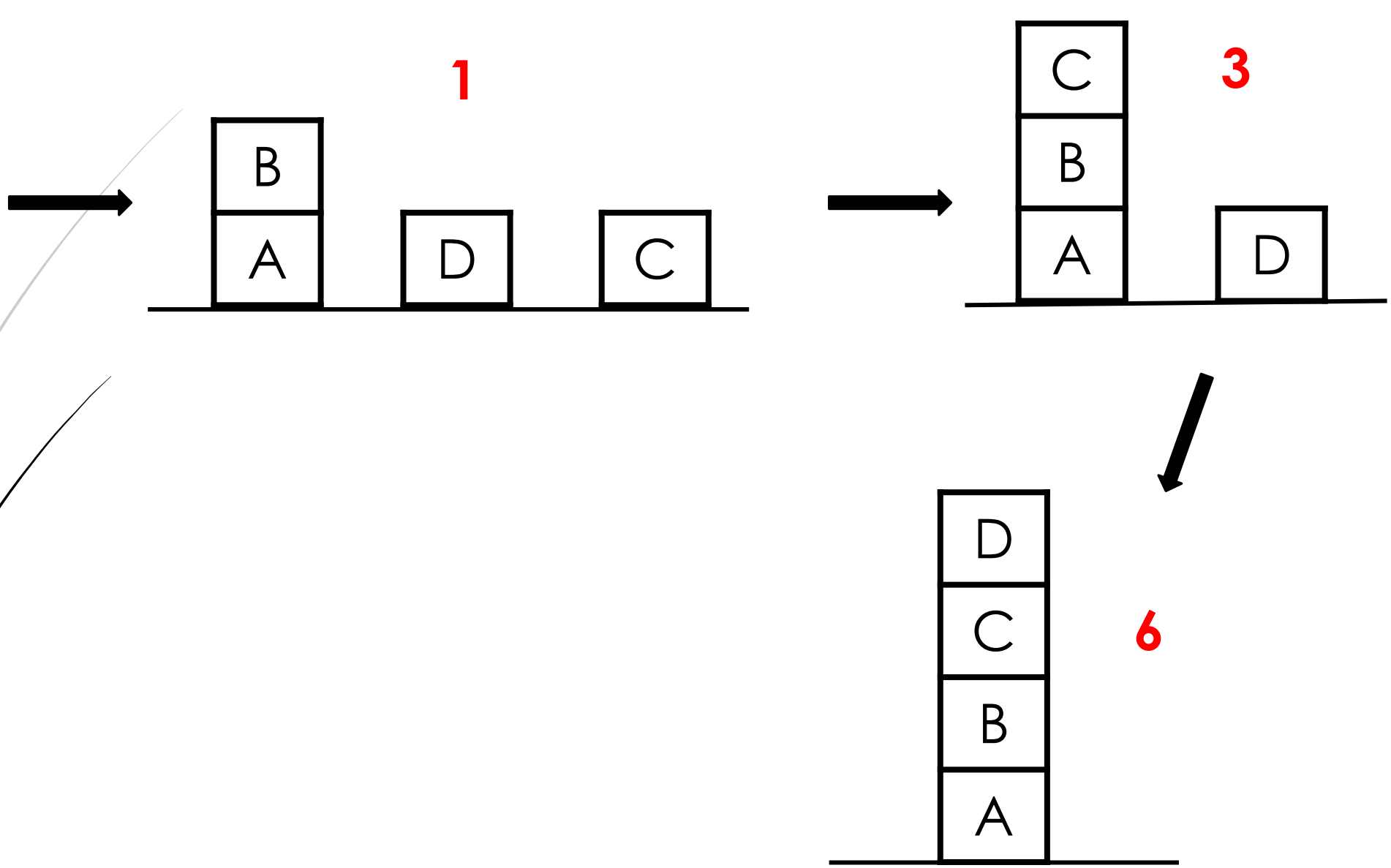
Goal State = 6



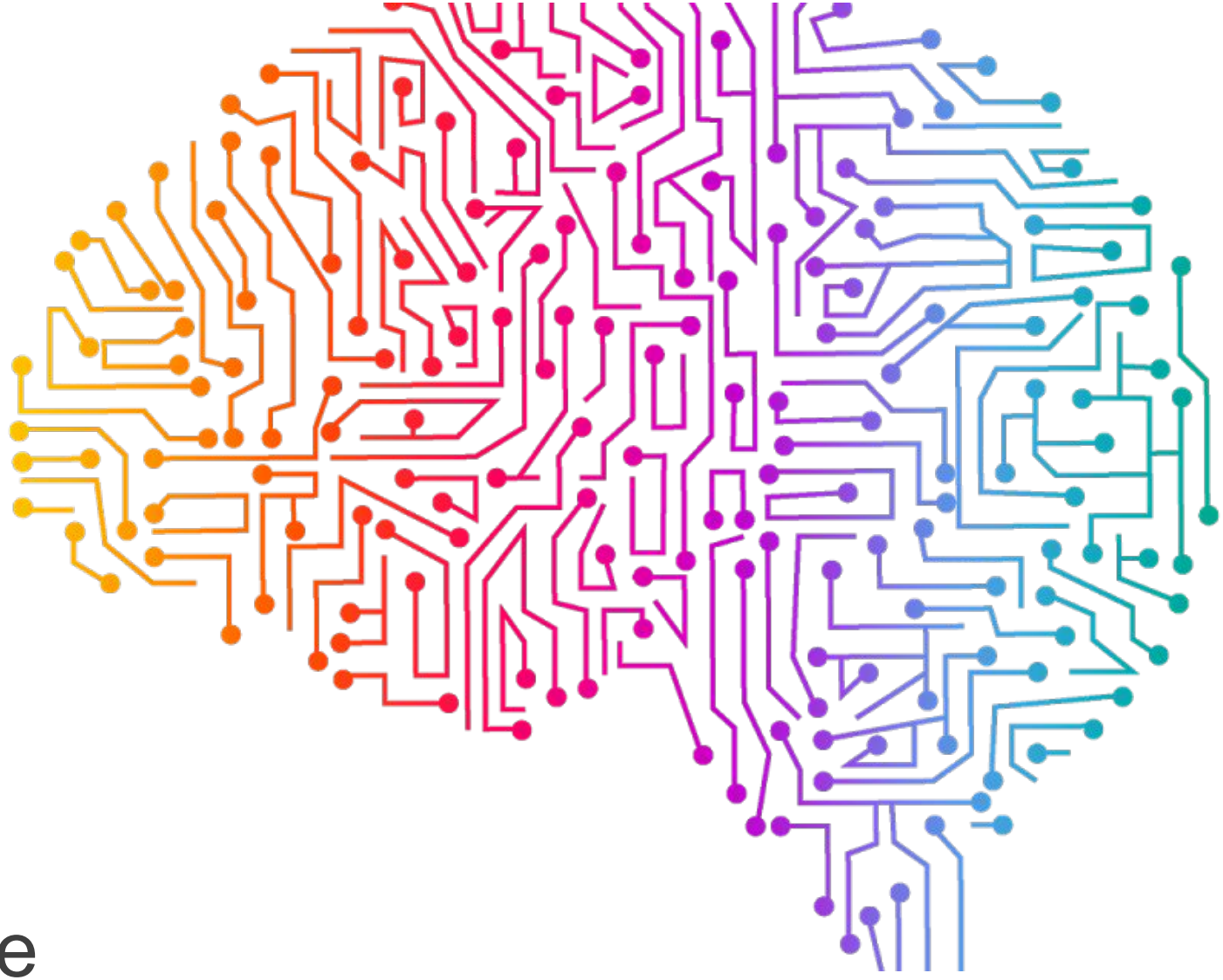
Local Heuristic:

- for each block that has the correct support structure: +1 to every block in the support structure
- for each block that has the wrong support structure: -1 to every block in the support structure
- Operator: Bring any block to the ground or on another block


-6
A
D
C
B
D
C
B
A
-3
-1
C
B
A
D
0
B
A
D
C


1
B
A
D
C
3
C
B
A
D
6
D
C
B
A
Goal State

Advantages and Disadvantages of Hill Climbing (with Real-Life Applications)



Advantages and Disadvantages

Advantages:

- It is a very useful technique while solving problems like job searching, shopping, web exploring for a certain query, and management related tasks etc.
- It requires very less computational power.
- The agent moves in the direction of the goal which optimizes our cost.

Disadvantages:

- The efficiency and effectiveness get compromised while using this technique.
- If the value of the heuristic is uncertain then this technique is not recommended.
- It is an immediate solution, not an effective solution.
- The results obtained from this technique are uncertain and are not reliable.


Questions
?
Answers
?
?