21CS54 – Artificial Intelligence and Machine Learning

Module 2

Informed Search Strategies

Text Book: Stuart J. Russell and Peter Norvig, Artificial Intelligence, 3rd Edition, Pearson,2015

Arpit Sharma
Department of Computer Science and Engineering
CMR Institute of Technology

Overview of Lesson Plan

- Module – 2
  - Informed Search Strategies
    - Greedy best-first search
    - A*search
    - Heuristic Functions
  - Introduction to Machine Learning
  - Understanding Data

Informed Search Strategies

- It is also known as heuristic search, involves using domain-specific knowledge or heuristics to guide the search process towards the goal state.
- **Key Components**:

  a. Heuristics: Domain-specific knowledge that estimates the cost or distance to the goal.

  b. Search Algorithms: Utilize heuristics to make informed decisions during the search process.

Greedy best-first search

- Greedy Best-First Search is a heuristic-based search algorithm used in artificial intelligence for navigating through a search space towards a goal.
- It's a part of informed search algorithms that employ heuristic functions to guide the search.

Basic Concept:

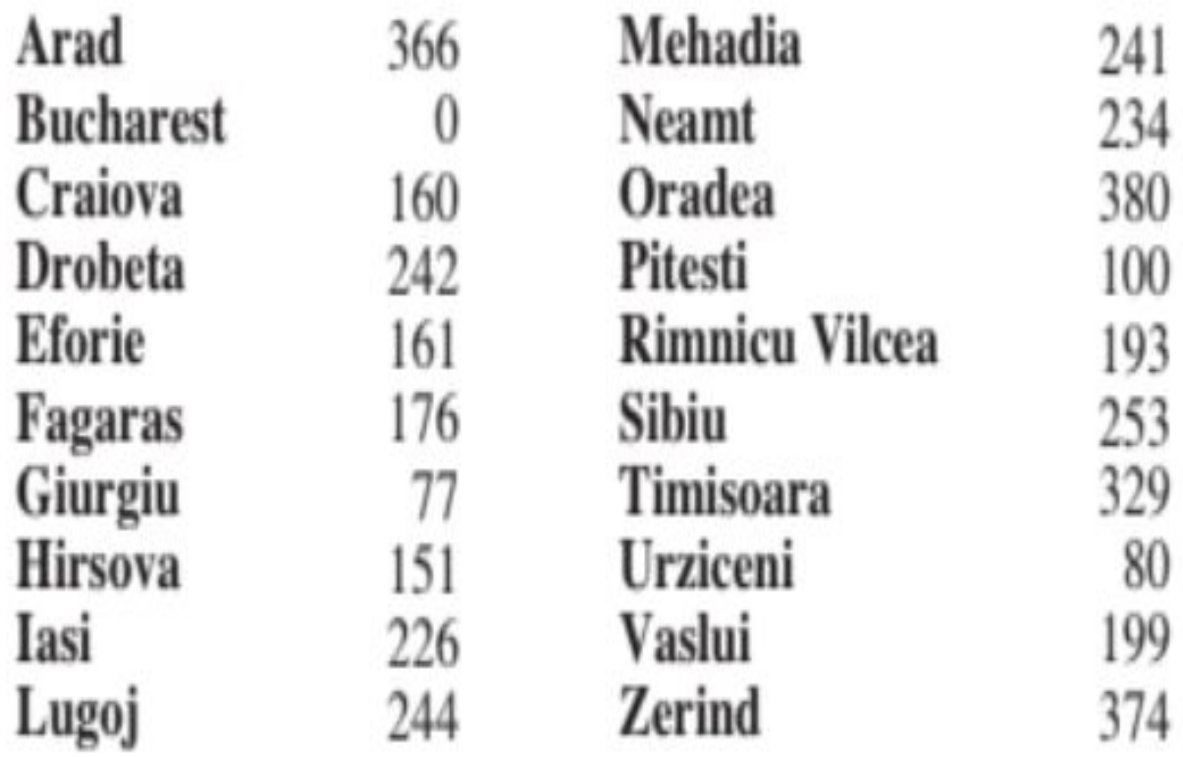
- GBFS selects the node that appears to be the most promising at any given moment, based solely on the heuristic information available.
- It prioritizes nodes for expansion based on a heuristic evaluation, moving towards the node that seems closest to the goal according to the heuristic function.
- Unlike other search algorithms, GBFS doesn’t consider the total cost or distance from the start but focuses on reaching the goal quickly based on heuristic estimates.

$$F(n)=h(n)$$

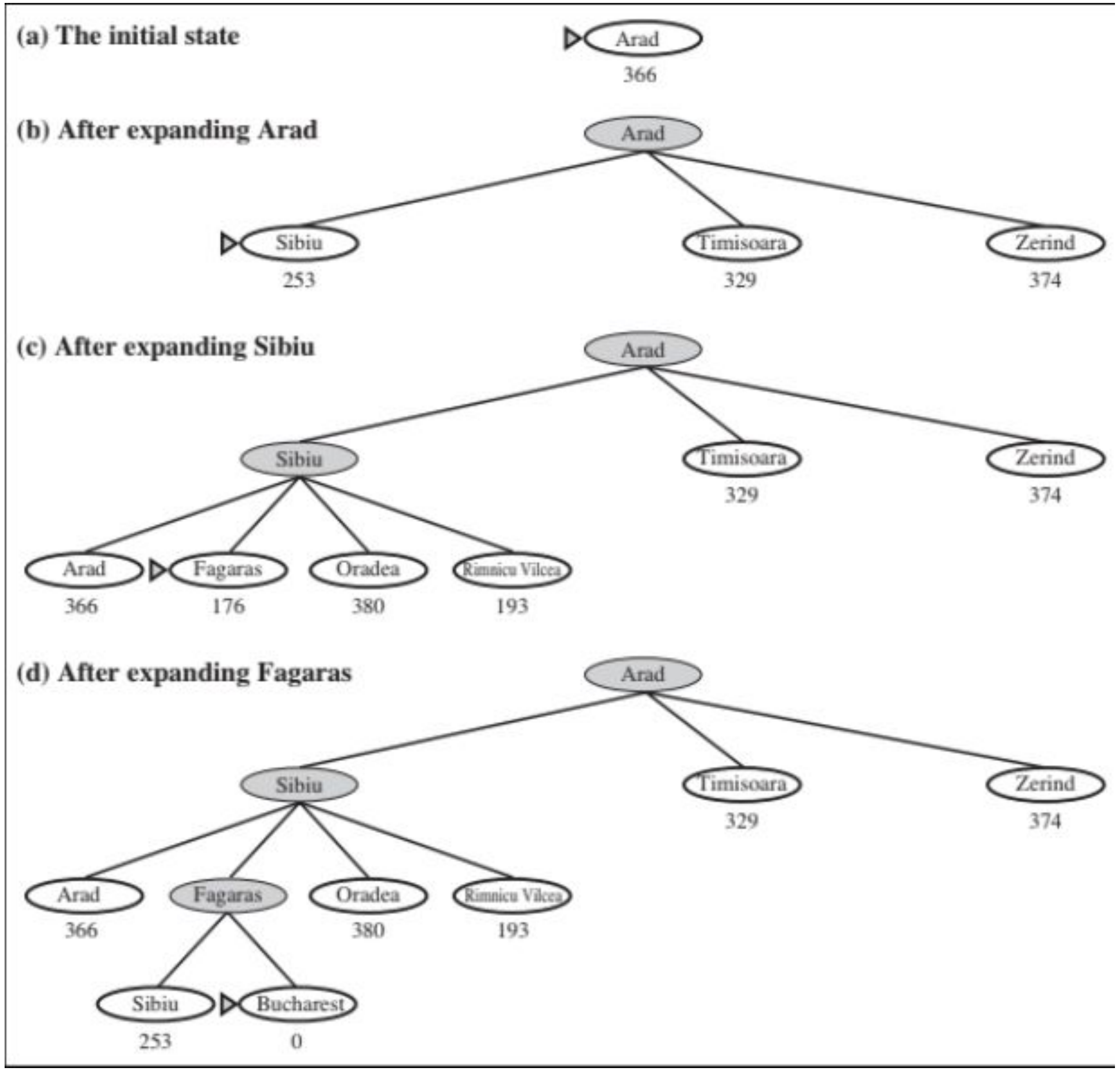
where h(n) is the heuristic function.

- A heuristic function is a technique used in problem-solving and decision-making that aims to efficiently find approximate solutions when an exact solution is either impossible, impractical, or computationally expensive to obtain.

Heuristic Values



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

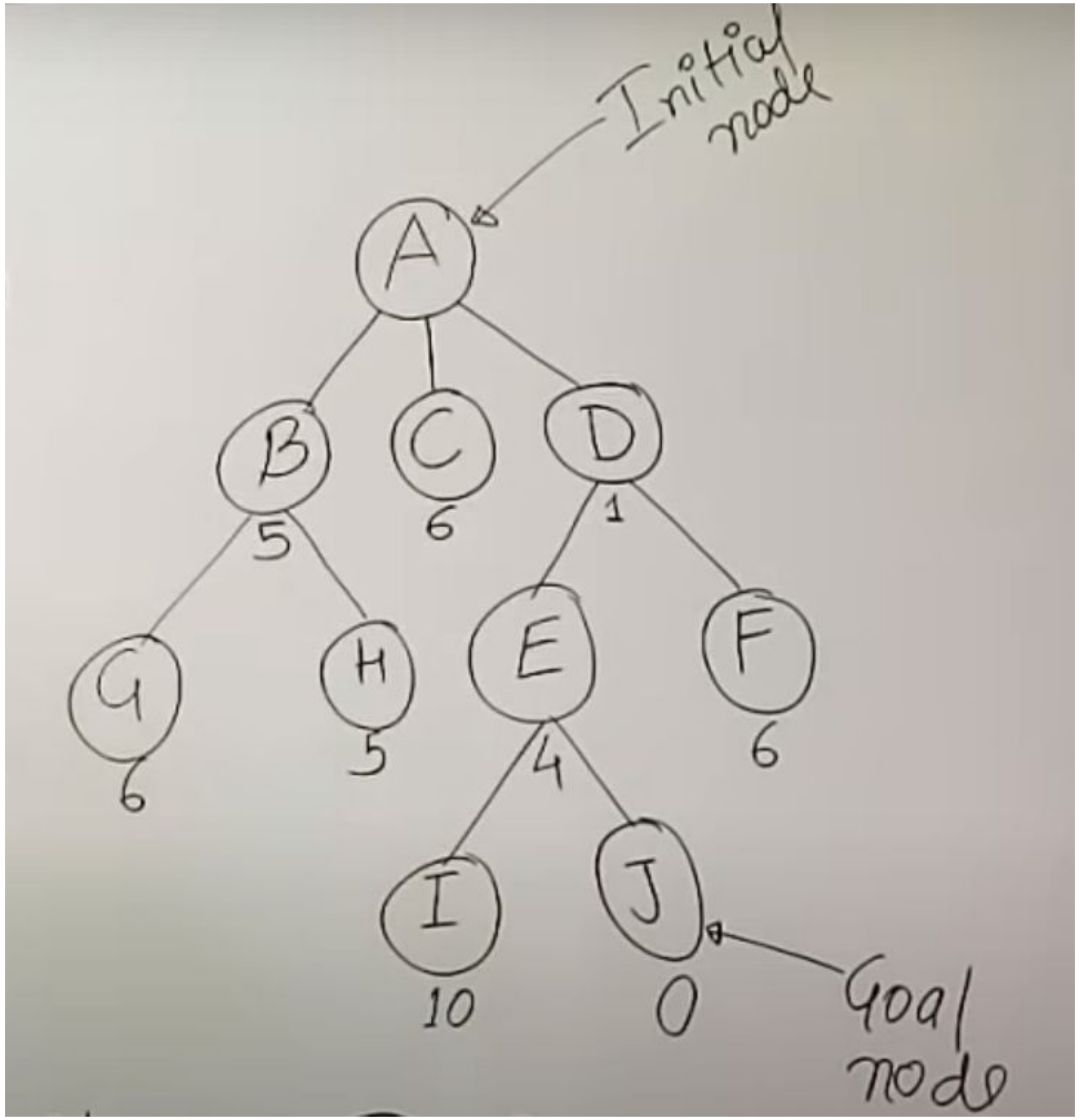

(a) The initial state
Arad
366
(b) After expanding Arad
Arad
Sibiu
253
Timisoara
329
Zerind
374
(c) After expanding Sibiu
Arad
Sibiu
Timisoara
329
Zerind
374
Arad
366
Fagaras
176
Oradea
380
Rimnicu Vilcea
193
(d) After expanding Fagaras
Arad
Sibiu
Timisoara
329
Zerind
374
Arad
366
Fagaras
Oradea
380
Rimnicu Vilcea
193
Sibiu
253
Bucharest
0

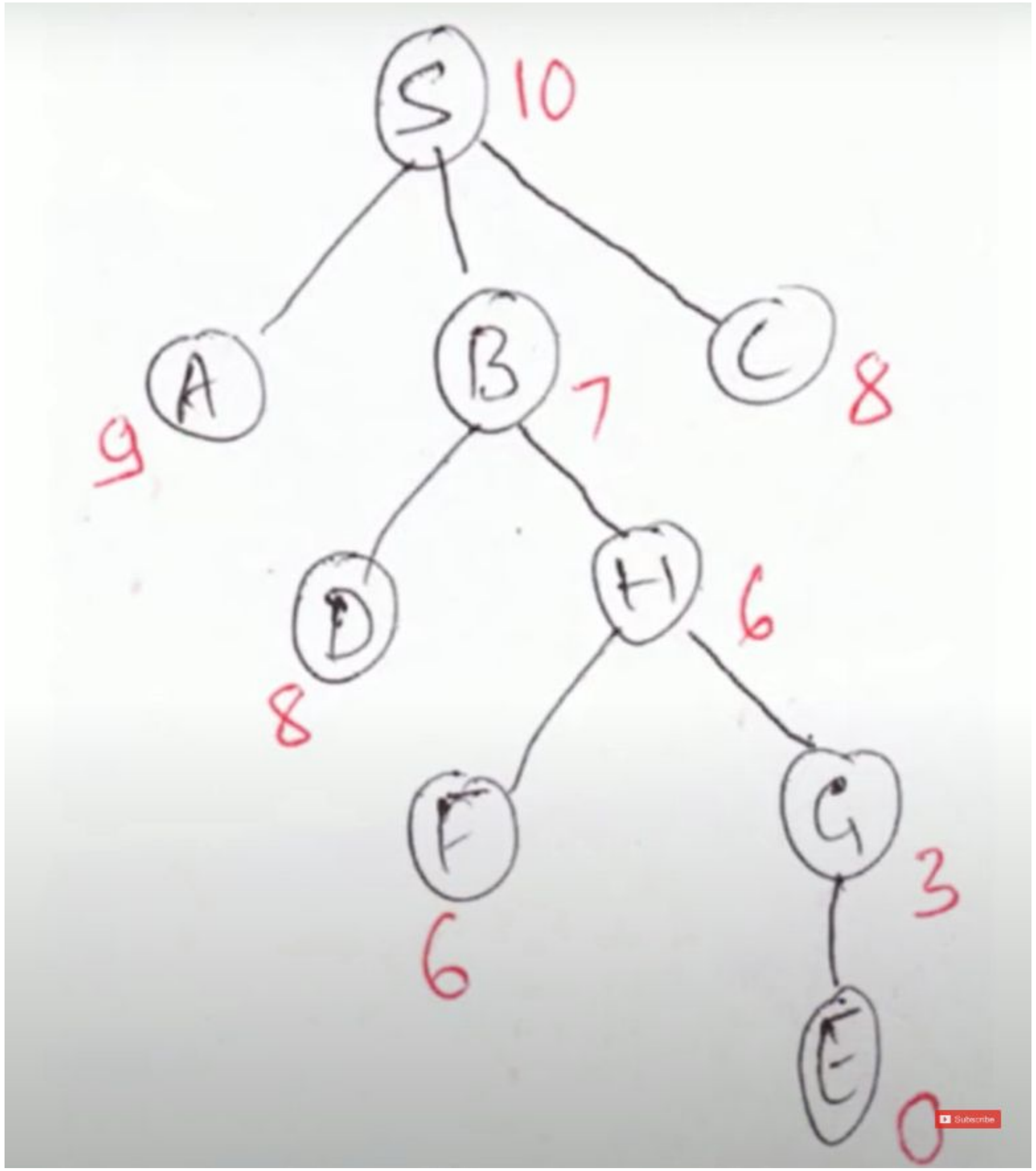
Initial State: Arab

Goal State: Bucharest

- **Time Complexity:** $O(b^d)$

Space Complexity: $O(b^d)$


Initial node
A
B
C
D
5
6
1
G
H
E
F
6
5
4
6
I
J
10
0
Goal node


S 10
A 9
B 7
C 8
D 8
H 6
F 6
G 3
E 0
Subscribe

Final Path: ADEJ

Advantages

1. **Quick Identification of Promising Paths:** GBFS swiftly identifies and explores paths that appear to lead closer to the goal based on the heuristic evaluation. This ability to quickly focus on promising paths makes it suitable for scenarios where finding a feasible solution fast is more crucial than finding the optimal one.
2. **Flexible Heuristic Usage:** GBFS allows for flexibility in using various heuristic functions. Depending on the problem domain, different heuristic functions can be employed to guide the search effectively towards the goal.
3. **Adaptability to Different Search Spaces:** GBFS can be adapted to different types of search spaces, including graphs, trees, or other problem representations, making it versatile in various problem-solving applications.

4. **Simplicity in Implementation:** Compared to some complex algorithms, GBFS is relatively straightforward to implement. Its simplicity makes it accessible and easier to apply in different scenarios.
5. **Effective in Certain Well-Structured Problems:** In problems where the heuristic provides accurate guidance towards the goal and there are no loops or cycles, GBFS can efficiently find solutions, especially if the heuristic closely matches the actual cost to reach the goal.

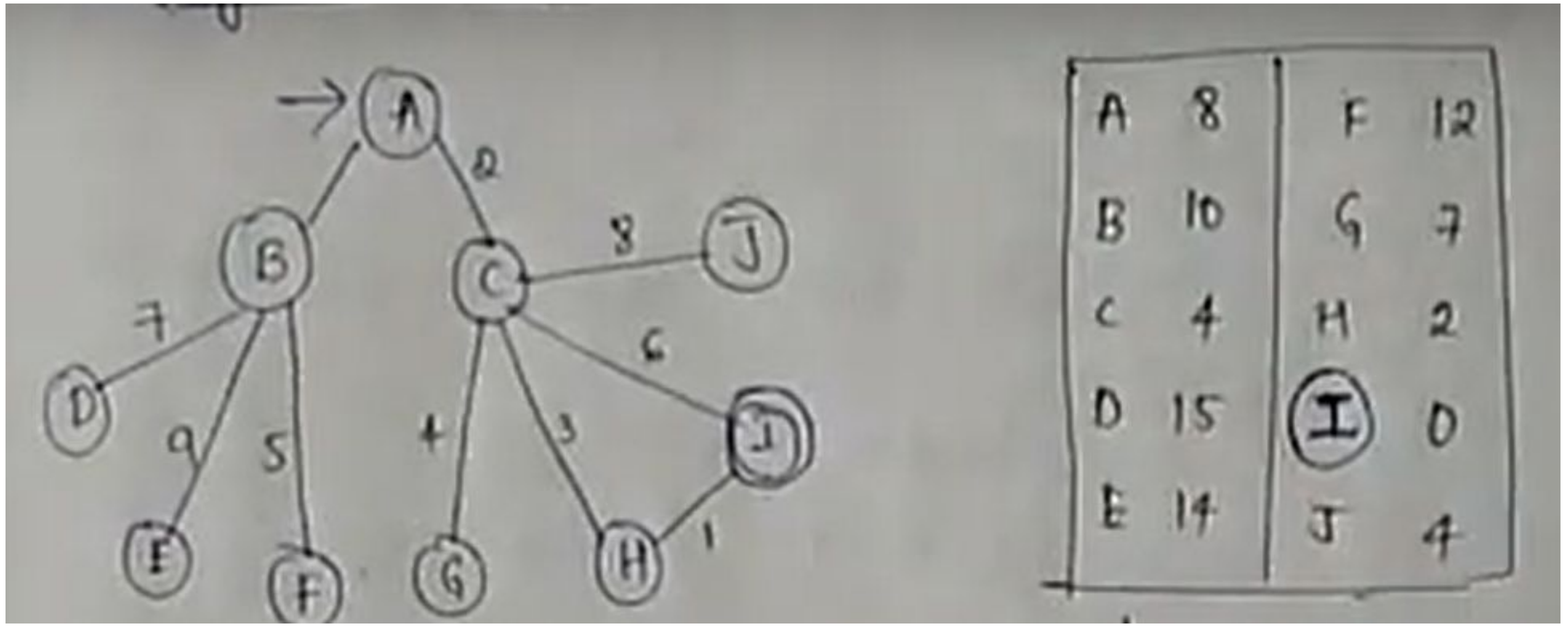
Challenges or Limitations

1. **Lack of Completeness**: GBFS is not guaranteed to find a solution if one exists. It might get stuck in loops or cycles, especially in scenarios where the heuristic function doesn't properly estimate the actual distance or cost to the goal.
2. **Non-Optimality**: GBFS does not always find the most optimal solution. It prioritizes nodes solely based on the heuristic value without considering the full path cost. This approach can lead to a suboptimal solution where the best possible path might not be explored due to the heuristic's bias.
3. **Heuristic Dependence**: The effectiveness of GBFS heavily relies on the accuracy of the heuristic function used. If the heuristic function is poorly designed or does not accurately estimate the distance to the goal, GBFS might fail to efficiently explore the search space.

4. **Memory Requirements**: GBFS might consume a significant amount of memory, particularly when dealing with large search spaces or when the branching factor (number of possible actions from each state) is high. This can make it impractical or inefficient in memory-constrained environments.

5. **Inability to backtrack**: GBFS doesn't possess backtracking capabilities. If a chosen path leads to a dead-end or an unsolvable state, GBFS won't backtrack to explore alternative paths, potentially missing the correct solution.

6. **Sensitivity to Initial Conditions**: The starting point or initial state in GBFS can significantly impact the solution quality. Depending on the initial node selected, GBFS might converge to different solutions, which may or may not be optimal.



A	8	F	12
B	10	G	7
C	4	H	2
D	15	I	0
E	14	J	4

A* Search

- The most widely known form of best-first search is called A∗ (pronounced "A-star ∗ search").
- It evaluates nodes by combining g(n), the cost to reach the node, and h(n), the cost

to get from the node to the goal:

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

Where,

g(n) gives the actual path cost from the start node to node n.

h(n) is the estimated cost of the cheapest path from n to the goal

f(n) is estimated cost of the cheapest solution through n .

Note: A* is both complete and optimal.

Conditions for optimality: Admissibility and consistency

Admissibility: The first condition we require for optimality is that h(n) be an admissible heuristic. An admissible heuristic is one that never overestimates the actual cost to reach the goal.

- In simpler terms, an admissible heuristic is optimistic; it never suggests a cost that is higher than the actual lowest cost to reach the goal.
- Using an admissible heuristic in A* ensures that the algorithm will find the optimal solution if one exists.

$$h(n) \leq h^*(n)$$

where,
h(n) – estimated cost from node to goal
h*(n)- actual cost to reach goal.

Consistency: A consistent (or monotonic) heuristic is a slightly stronger property than admissibility.

- It refers to a heuristic where the estimated cost from one state to another plus the estimated cost from that state to the goal is always greater than or equal to the estimated cost directly from the initial state to the goal.
- In other words, if the heuristic is consistent, it maintains a certain relationship between the cost estimates of different states in such a way that it guarantees the optimality of the A* algorithm.

$$h(A) \leq g(A, B) + h(B)$$

Where,

H(A) – estimated cost from node A to the goal.

g(A,B)- actual cost from node A to B

H(B)- estimated cost from node B to goal.

For example, if you're finding a path from node A to node C using A*, and the heuristic function calculates the estimated cost from A to Goal (h(A)) as 10, while the known cost from A to B (g(A, B)) is 6 and from B to Goal h(B) is 5, the triangle inequality holds:

$h(A) \leq g(A, B) + h(B)$

$10 \leq 6 + 5$

$10 \leq 11$

Therefore, the heuristic function would be consistent in this scenario for the A* algorithm.

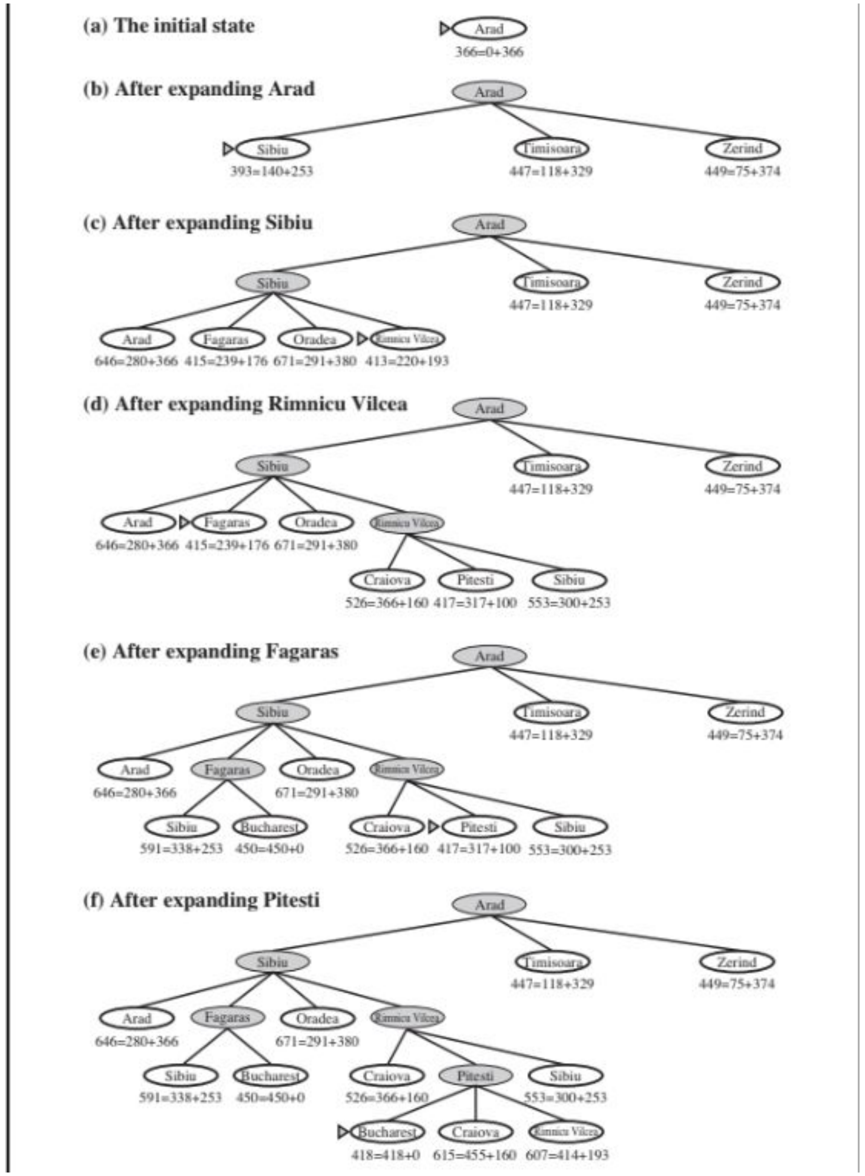
Optimality of A*

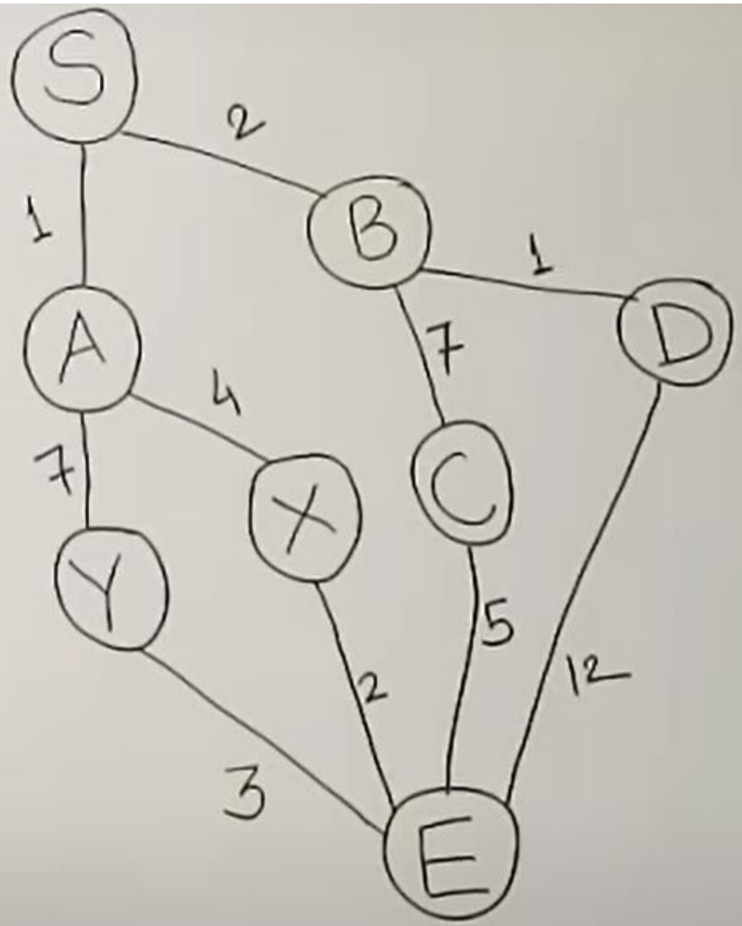
A∗ has the following properties:

- The tree-search version of A ∗ is optimal if h(n) is admissible, while the graph-search version is optimal if h(n) is consistent.

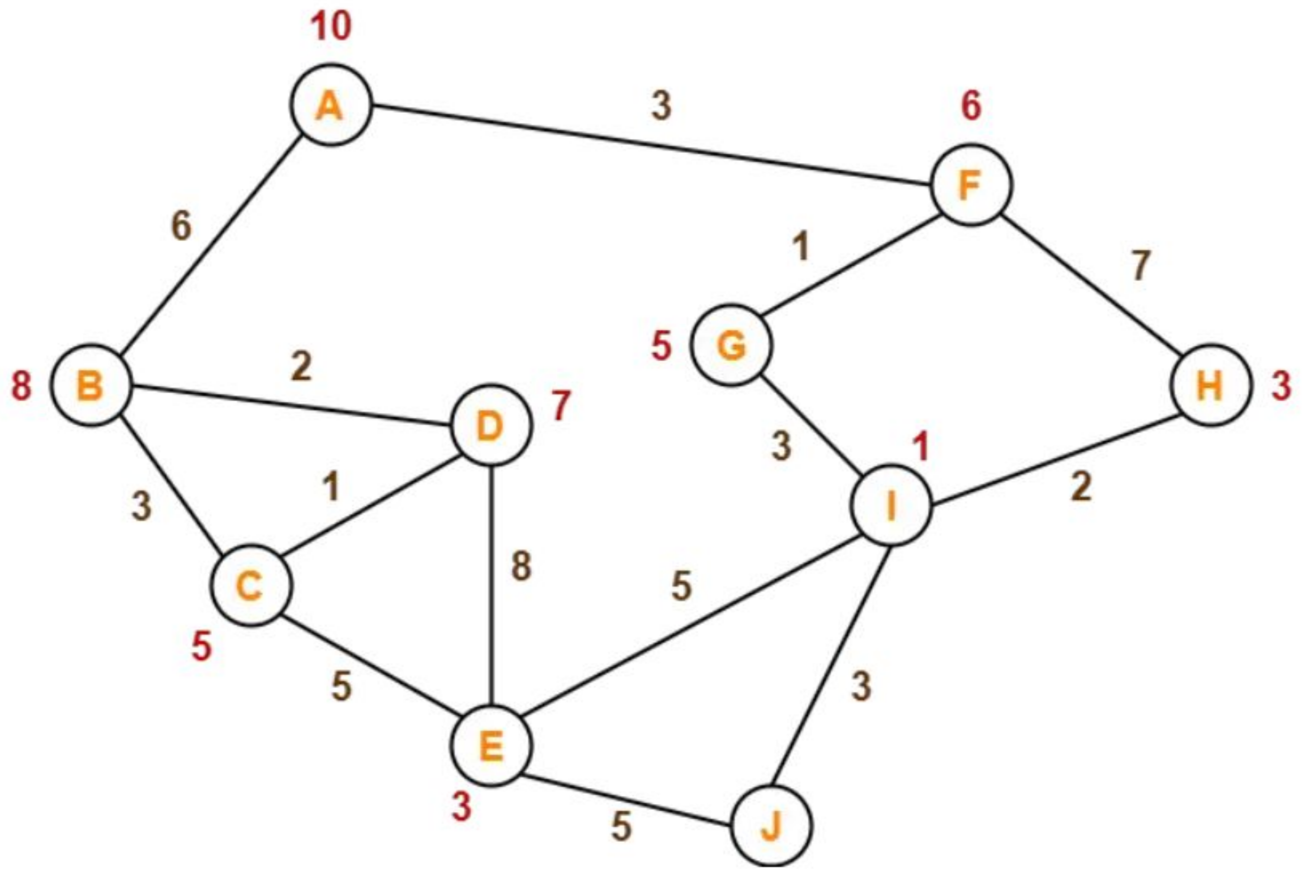
We show that the graph-search version is optimal

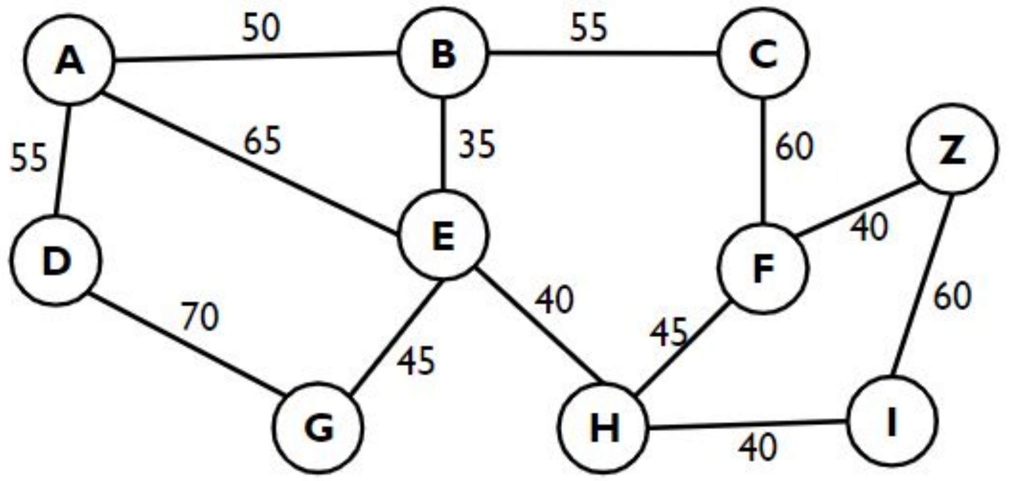
- if h(n) is consistent, then the values of f(n) along any path are nondecreasing.
- The proof follows directly from the definition of consistency.


(a) The initial state
Arad
366=0+366
(b) After expanding Arad
Arad
Sibiu 393=140+253
Timisoara 447=118+329
Zerind 449=75+374
(c) After expanding Sibiu
Arad
Sibiu
Timisoara 447=118+329
Zerind 449=75+374
Arad 646=280+366
Fagaras 415=239+176
Oradea 671=291+380
Rimnicu Vilcea 413=220+193
(d) After expanding Rimnicu Vilcea
Arad
Sibiu
Timisoara 447=118+329
Zerind 449=75+374
Arad 646=280+366
Fagaras 415=239+176
Oradea 671=291+380
Rimnicu Vilcea
Craiova 526=366+160
Pitesti 417=317+100
Sibiu 553=300+253
(e) After expanding Fagaras
Arad
Sibiu
Timisoara 447=118+329
Zerind 449=75+374
Arad 646=280+366
Fagaras
Oradea 671=291+380
Rimnicu Vilcea
Sibiu 591=338+253
Bucharest 450=450+0
Craiova 526=366+160
Pitesti 417=317+100
Sibiu 553=300+253
(f) After expanding Pitesti
Arad
Sibiu
Timisoara 447=118+329
Zerind 449=75+374
Arad 646=280+366
Fagaras
Oradea 671=291+380
Rimnicu Vilcea
Sibiu 591=338+253
Bucharest 450=450+0
Craiova 526=366+160
Pitesti
Sibiu 553=300+253
Bucharest 418=418+0
Craiova 615=455+160
Rimnicu Vilcea 607=414+193



$A \rightarrow 5$	$X \rightarrow 5$
$B \rightarrow 6$	$Y \rightarrow 8$
$C \rightarrow 4$	
$D \rightarrow 15$	


10
A
3
6
F
6
1
7
G
5
8
B
2
D
7
H
3
3
1
I
1
2
C
1
8
5
5
3
5
E
3
5
J



Location	Straight line distance to **Z**	Location	Straight line distance to **Z**
A	100	F	30
B	70	G	110
C	30	H	70
D	120	I	50
E	90		

HEURISTIC FUNCTIONS

Heuristics for the 8-puzzle

Recall that the object of the puzzle is to slide the tiles horizontally or vertically into the empty space until the configuration matches the goal configuration.

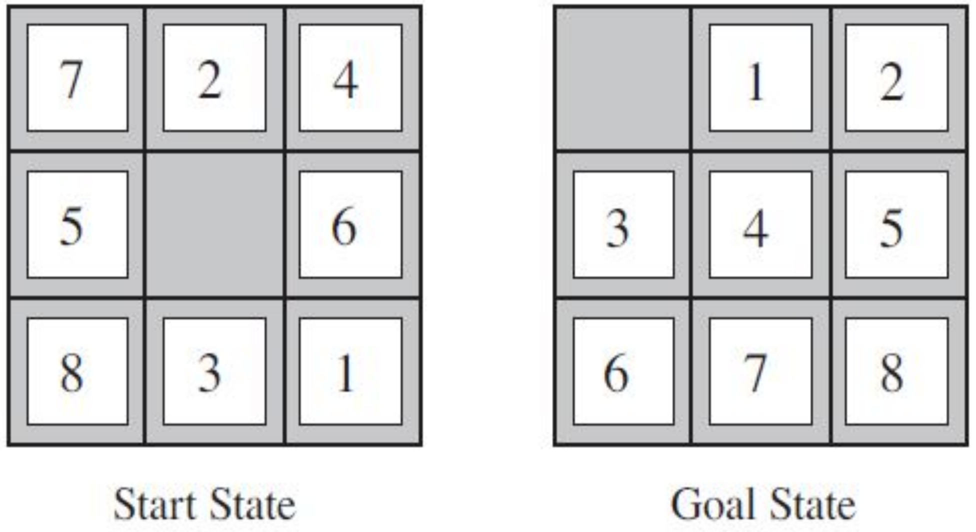


Figure 3.28 A typical instance of the 8-puzzle. The solution is 26 steps long.

Two commonly used heuristic for 8 puzzle

Candidate 1

h1 = the number of misplaced tiles.

For Figure 3.28, all of the eight tiles are out of position, so the start state would have h1 = 8.

h1 is an admissible heuristic because it is clear that any tile that is out of place must be moved at least once.

Candidate 2

h2 = the sum of the distances of the tiles from their goal positions.

Because tiles cannot move along diagonals, the distance we will count is the sum of the horizontal and vertical distances.

This is sometimes called the **city block distance** or **Manhattan distance**.

h2 is also admissible because any move can do is move one tile one step closer to the goal.

Tiles 1 to 8 in the start state give a Manhattan distance of

h2 = 3+1 + 2 + 2+ 2 + 3+ 3 + 2 = 18 .

As expected, neither h1 nor h2 overestimates the true solution cost, which is 26.