

Naive Bayes

In a Naive Bayes classifier, if the probability of Class A is 0.6, the probability of Class B is 0.4, and the likelihood of a feature F given Class A is 0.2 while given Class B is 0.3, what is the posterior probability of Class A given feature F ?

Naive Bayes

Given the following data for a Naive Bayes classifier:

$$P(\text{Class A}) = 0.6$$

$$P(\text{Class B}) = 0.4$$

$$P(\text{Feature1}|\text{Class A}) = 0.3$$

$$P(\text{Feature1}|\text{Class B}) = 0.5$$

What is the posterior probability $P(\text{Class A}|\text{Feature1})$ assuming no other features?

Search Algorithm

Uninformed Strategies use only the information available in the problem definition.

1. Breadth First Search (BFS)

- Explores all nodes at the present depth level before moving on to the next level.
- Guarantees finding the shortest path in unweighted graphs.
- Uses a queue for storing nodes to be explored.

2. Depth First Search (DFS)

- Explores as far down a branch as possible before backtracking.
- Can be implemented using a stack.
- Less memory-intensive but may not find the shortest path and can get stuck in deep branches.

3. Uniform Cost Search (UCS)

- Expands the node with the lowest path cost (the total cost from the start node to the current node).
- Useful for weighted graphs where costs vary between edges.
- In a situation where all edge costs are equal, Uniform Cost Search behaves like Breadth-First Search
- If Uniform Cost Search encounters a node that has already been expanded, It will ignore that node.

Search Algorithm

Informed Strategies:

These algorithms utilize heuristic information to guide the search process.

The purpose of the heuristic function $h(n)$ is to estimate the cost from the current node to the goal.

1. Greedy Search

- Expands the node that appears closest to the goal based on a heuristic, without considering the cost to reach that node.
- Can be faster but does not guarantee the optimal solution.

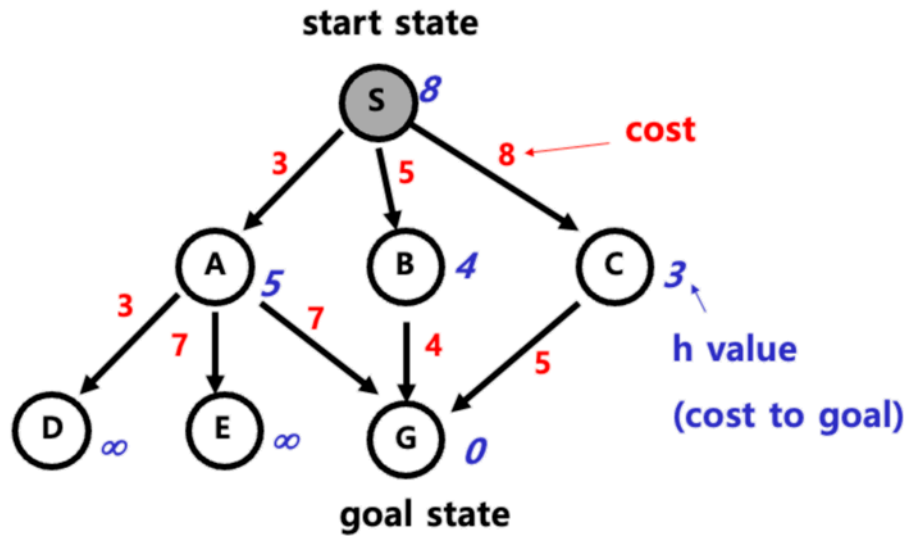
2. A* Search

- Combines the cost to reach a node and an estimate of the cost to reach the goal.
- Uses a priority queue to explore nodes

Search Algorithm

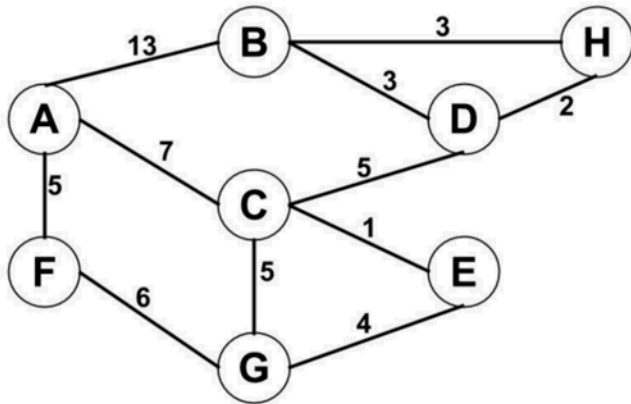
Find a path from S to G using the following search algorithms:

- Breadth First Search
- Depth First Search
- Uniform Cost Search (calculate cost)
- A* Search



Search Algorithm

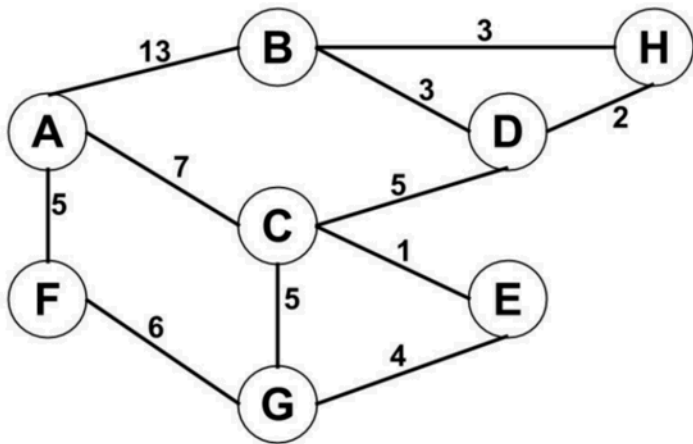
1. Find an optimal path from A to H using Uniform Cost Search
2. Find the path between A to H using Greedy Search
3. Find the shortest path between A to H using A* Search algorithm



From	Straight line distance
A	14
B	3
C	7
D	2
E	6
F	10
G	11
H	0

Search Algorithm

- **Uniform Cost Search:** Expand least-cost unexpanded node

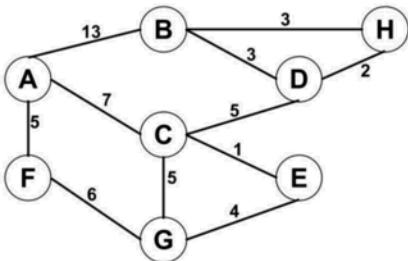


Search Algorithm

- **Greedy Search:** Expand the node that appears to be closest to goal

Evaluation function $f(n) = h(n)$

$h(n)$ = estimated cost from n to goal



From	Straight line distance
A	14
B	3
C	7
D	2
E	6
F	10
G	11
H	0

Search Algorithm

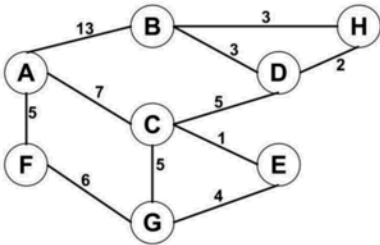
- **A* Search:** Expand the node that minimize the estimated total cost of path through n to goal

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

$g(n)$ = cost to reach node n

$h(n)$ = estimated cost from n to goal

$f(n)$ = estimated total cost of path through n to goal



From	Straight line distance
A	14
B	3
C	7
D	2
E	6
F	10
G	11
H	0