Heuristic Best-First Search Algoritm

What is the Greedy-Best-first search algorithm?

> Greedy Best-First Search is an AI search algorithm that attempts to find the most promising path from a given starting point to a goal. It prioritizes paths that appear to be the most promising, regardless of whether or not they are actually the shortest path. The algorithm works by evaluating the cost of each possible path and then expanding the path with the lowest cost. This process is repeated until the goal is reached.

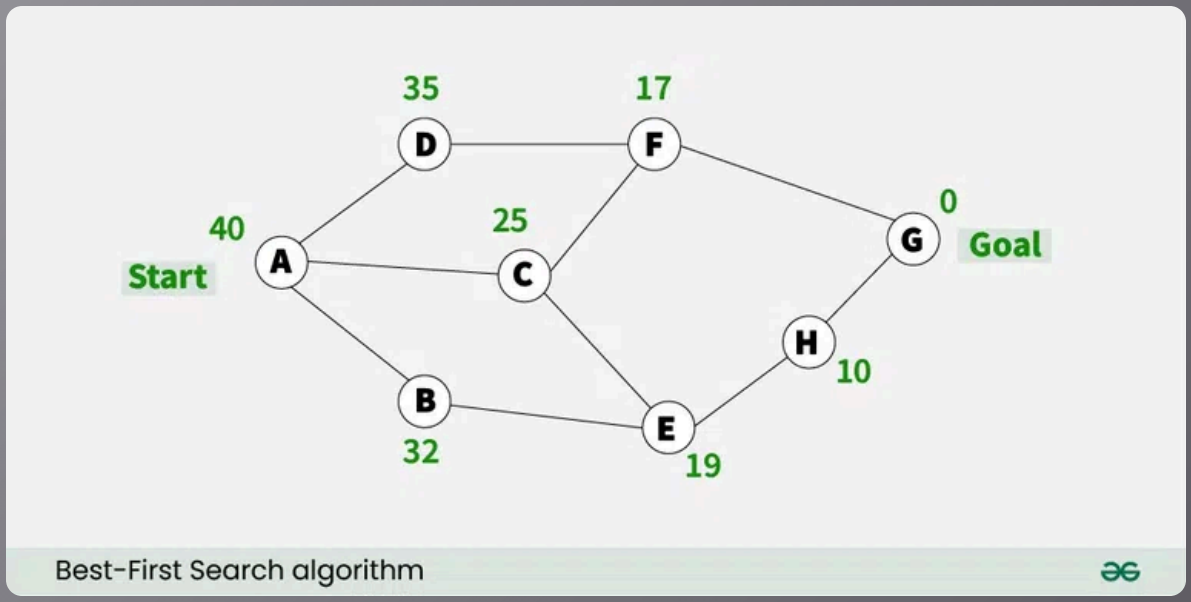
The algorithm works by using a heuristic function to determine which path is the most promising. The heuristic function takes into account the cost of the current path and the estimated cost of the remaining paths. If the cost of the current path is lower than the estimated cost of the remaining paths, then the current path is chosen. This process is repeated until the goal is reached.

How Greedy Best-First Search Works?

- Greedy Best-First Search works by evaluating the cost of each possible path and then expanding the path with the lowest cost. This process is repeated until the goal is reached.
- The algorithm uses a heuristic function to determine which path is the most promising.
- The heuristic function takes into account the cost of the current path and the estimated cost of the remaining paths.
- If the cost of the current path is lower than the estimated cost of the remaining paths, then the current path is chosen. This process is repeated until the goal is reached.

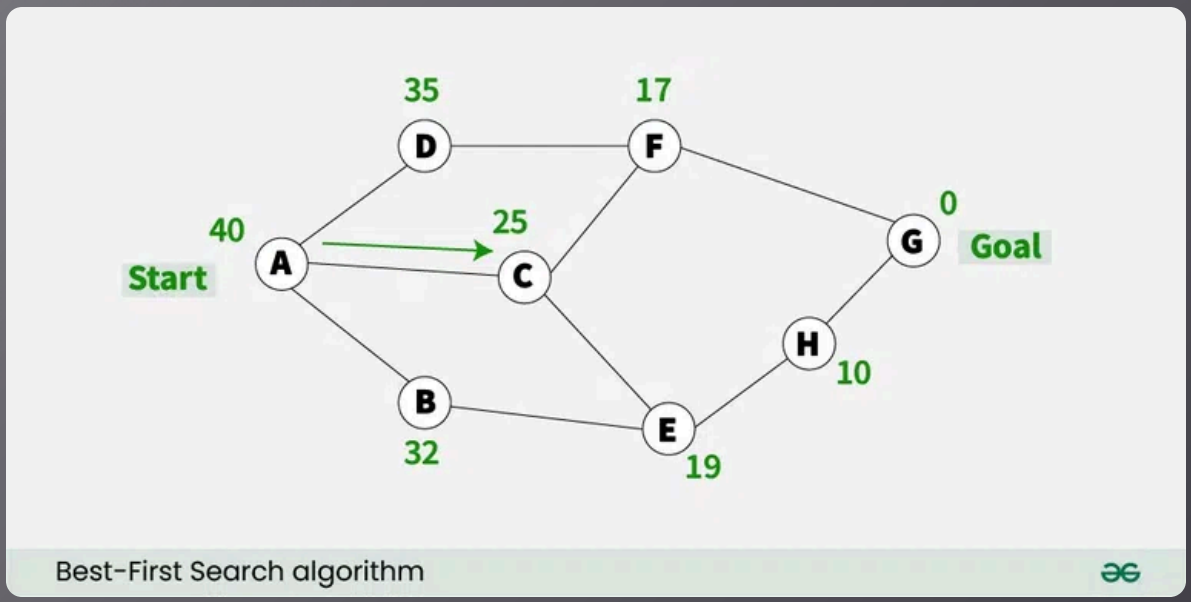
An example of the best-first search algorithm is below graph, suppose we have to find the path from A to G

!! **The values in red color represent the heuristic value of reaching the goal node G from current node**

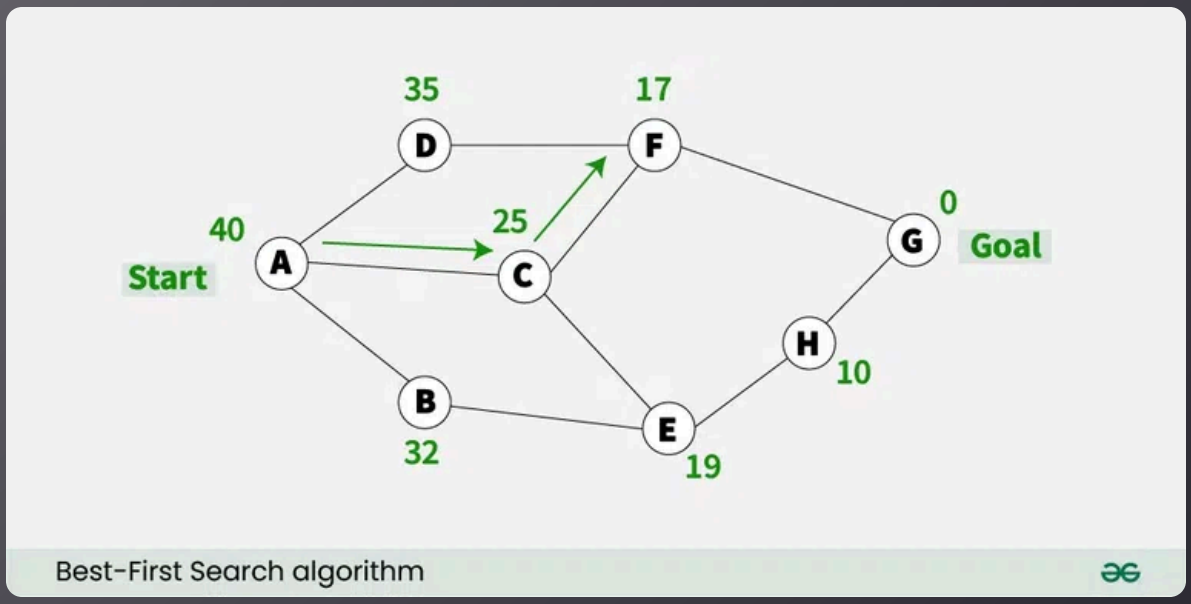


!! 1. We are starting from A , so from A there are direct path to node B(with heuristics value of 32) , from A to C (with heuristics value of 25) and from A to D(with heuristics value of 35) .

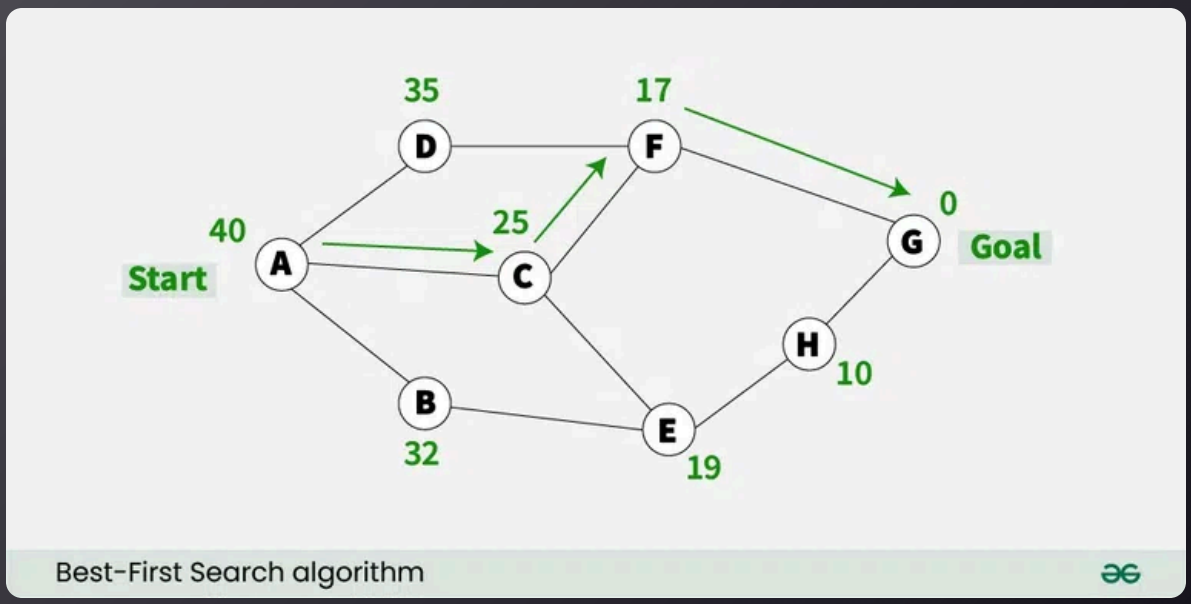
!! 2. So as per best first search algorithm choose the path with lowest heuristics value , currently C has lowest value among above node . So we will go from A to C.



3. Now from C we have direct paths as C to F(with heuristics value of 17) and C to E(with heuristics value of 19) , so we will go from C to F.



4. Now from F we have direct path to go to the goal node G (with heuristics value of 0) , so we will go from F to G.



5. So now the goal node G has been reached and the path we will follow is **A->C->F->G** .

Advantages of Greedy Best-First Search:

- **Simple and Easy to Implement:** Greedy Best-First Search is a relatively straightforward algorithm, making it easy to implement.
- **Fast and Efficient:** Greedy Best-First Search is a very fast algorithm, making it ideal for applications where speed is essential.
- **Low Memory Requirements:** Greedy Best-First Search requires only a small amount of memory, making it suitable for applications with limited memory.
- **Flexible:** Greedy Best-First Search can be adapted to different types of problems and can be easily extended to more complex problems.
- **Efficiency:** If the heuristic function used in Greedy Best-First Search is good to estimate, how close a node is to the solution, this algorithm can be a very efficient and find a solution quickly, even in large search spaces.

Disadvantages of Greedy Best-First Search:

- **Inaccurate Results:** Greedy Best-First Search is not always guaranteed to find the optimal solution, as it is only concerned with finding the most promising path.
- **Local Optima:** Greedy Best-First Search can get stuck in local optima, meaning that the path chosen may not be the best possible path.
- **Heuristic Function:** Greedy Best-First Search requires a heuristic function in order to work, which adds complexity to the algorithm**.**
- **Lack of Completeness:** Greedy Best-First Search is not a complete algorithm, meaning it may not always find a solution if one is exists. This can happen if the algorithm gets stuck in a cycle or if the search space is a too much complex.

Applications of Greedy Best-First Search:

- **Pathfinding:** Greedy Best-First Search is used to find the shortest path between two points in a graph. It is used in many applications such as video games, robotics, and navigation systems.
- **Machine Learning:** Greedy Best-First Search can be used in machine learning algorithms to find the most promising path through a search space.
- **Optimization:** Greedy Best-First Search can be used to optimize the parameters of a system in order to achieve the desired result.

- **Game AI:** Greedy Best-First Search can be used in game AI to evaluate potential moves and chose the best one.
- **Navigation:** Greedy Best-First Search can be use to navigate to find the shortest path between two locations.
- **Natural Language Processing:** Greedy Best-First Search can be use in natural language processing tasks such as language translation or speech recognisation to generate the most likely sequence of words.
- **Image Processing:** Greedy Best-First Search can be use in image processing to segment image into regions of interest.

Implementation in Python

```
import heapq

class Graph:
    def __init__(self):
        self.graph = {}
        self.heuristic = {}

    def add_edge(self, u, v):
        if u not in self.graph:
            self.graph[u] = []
        self.graph[u].append(v)

    def set_huristic(self, h_values):
        self.heuristic = h_values


def greedy_bfs(self, start, goal):
        open_list = []
        heapq.heappush(open_list, (self.heuristic[start], start))
        visited = set()

        while open_list:
            _, node = heapq.heappop(open_list)
            if node in visited:
                continue
```

```
            print(f"Visiting: {node}")

            if node == goal:
                print("Goal reached!")
                return
            visited.add(node)


            for neighbor in self.graph.get(node, []):
                if neighbor not in visited:
                    heapq.heappush(open_list, (self.heuristic[neighbor],
neighbor))

        print("Goal not reachable!")



g = Graph()
g.add_edge('S', 'A')
g.add_edge('S', 'B')
g.add_edge('A', 'C')
g.add_edge('A', 'D')
g.add_edge('B', 'G')

heuristics = {'S': 6, 'A': 4, 'B': 2, 'C': 3, 'D': 5, 'G': 0}
g.set_heuristic(heuristics)
print("Path using Greedy Best-First Search:")
g.greedy_bfs('S', 'G')
```